PROJETO DE GRADUAÇÃO

OTIMIZAÇÃO DAS ROTAS LOGÍSTICAS DE CAMINHÕES DE COLETA SELETIVA A PARTIR DO USO DE INTERNET DAS COISAS: UM ESTUDO DE CASO

Por,

Gustavo Pessoa Caixeta Pinto da Luz

Brasília, 03 de novembro de 2021.

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia
Departamento de Engenharia de Produção

PROJETO DE GRADUAÇÃO

OTIMIZAÇÃO DAS ROTAS LOGÍSTICAS DE CAMINHÕES DE COLETA SELETIVA A PARTIR DO USO DE INTERNET DAS COISAS: UM ESTUDO DE CASO

Por,

Gustavo Pessoa Caixeta Pinto da Luz

Relatório submetido como requisito parcial para
obtenção do grau de Engenheiro de Produção

Banca Examinadora

Prof. Dr. Paulo Celso dos Reis Gomes, UnB/EPR (Orientador)

Prof. Dr. Reinaldo Crispiniano Garcia, UnB/EPR (Examinador)

Brasília-DF, 03 de novembro de 2021.

"A ciência é, portanto, uma perversão de si mesma, a menos que tenha como fim último, melhorar a humanidade."

Nikola Tesla (1856-1943), inventor sérvio-americano

AGRADECIMENTOS

Agradeço a todos que já me ajudaram a me tornar uma pessoa e profissional melhor. Primeiramente à minha família, que é a minha base de tudo e sempre apoiou as minhas decisões e me ensinou a ser quem eu sou. Ao meu pai Marcelo que sempre foi minha referência sobre estudar e aprender o máximo sobre assuntos julgados como complicados pela sociedade, à minha mãe Virgínia por sempre querer o meu melhor e à minha irmã Luciana por todo o companheirismo e ensinamentos sobre o que é compartilhar.

Aos meus avós maternos e paternos que sempre me deram todo o carinho possível e pela sorte de ser o único neto dos dois lados e conviver com todos enquanto vivos e morando na mesma cidade. Às minhas primas e tias que sempre foram amigas e presentes na minha vida, independentemente de onde estavam.

Agradeço aos "produmigos" que tornaram o curso muito mais prazeroso, não conseguiria sem vocês. Em especial à Isabela, "produmiga" e "produnamorada" que sempre me apoiou na vida e na graduação até o final. Também agradeço aos meus amigos do peito de fora da faculdade.

Aos professores excelentes que tive o prazer de ser aluno, obrigado por acreditarem na educação no Brasil e dedicarem suas vidas para que ela melhore. Ao laboratório UIoT, em especial ao Francisco e a equipe de *hardware* que me aceitaram como pesquisador da área mesmo eu sendo de uma formação diferente da tradicional e pelos ensinamentos de eletrônica, redes e programação.

Além disso, agradeço a todos os pesquisadores que trabalharam e produziram comigo, e aos que ainda acreditam no poder da pesquisa e ciência, que ainda não são valorizadas o necessário pela contribuição com a humanidade.

Por fim, agradeço ao *The Global Students SDG Challenge*, que possibilitou o surgimento do projeto que originou este trabalho, e a todos que se disponibilizaram para que a parceria internacional ocorresse.

RESUMO

O gerenciamento de resíduos e a alta concentração populacional são problemas que representam um grande desafio para a humanidade. A coleta seletiva surgiu para auxiliar nessas questões, porém ainda carece de muito investimento e melhorias, como o uso de tecnologia para reduzir custos e otimizar as operações. Nesse contexto de cidades inteligentes e a partir da Agenda 2030, surgiu na Dinamarca um projeto para criar uma solução que mede o nível do lixo em contêineres, que foi adaptada para o contexto brasileiro. O presente trabalho trata sobre a aplicação de um modelo de otimização de rotas logísticas em uma empresa de Brasília-DF a partir do uso de Internet das Coisas. Para isso, foi feita a comparação entre uma rota realizada pela empresa atualmente e a rota a ser realizada a partir do uso de sensores informando qual o preenchimento de cada contêiner. Utilizando a metodologia de mineração de dados CRISP-DM, foi possível entender mais sobre o negócio, coletar, preparar, filtrar e realizar a roteirização otimizada utilizando o *software* Google OR-Tools, seguindo a documentação do *Capacitated Vehicle Routing Problem*. Foram analisados fatores ambientais e financeiros, havendo a projeção dos ganhos ao longo do tempo. Os resultados mostraram a redução de 8,5% na distância gasta e 17% no tempo gasto por rota, o que traz a economia de mais de 40 mil reais e mais de 1,5 toneladas de dióxido de carbono despejadas na atmosfera por ano para a empresa analisada. Ainda há limitações como o número de dispositivos conectados enviando dados e número de rotas analisadas, porém foram trazidas informações de como continuar o projeto, que pode ser aplicado e personalizado de acordo com o contexto.

Palavras-chave: Roteirização; Coleta Seletiva; Internet das Coisas; Google OR-Tools; CRISP-DM.

ABSTRACT

Waste management and high population concentration are problems that represent a great challenge for humanity. Selective collection emerged to help with these issues, but it still lacks a lot of investment and improvements, such as the use of technology to reduce costs and optimize operations. In this context of smart cities and from the 2030 Agenda, a project emerged in Denmark to create a solution that measures the level of garbage in containers, which was adapted to the Brazilian context. This work deals with the application of a logistic route optimization model in a company in Brasília-DF, using Internet of Things. For this, a comparison was made between a route currently carried out by the company and the same route to be carried out using sensors informing the filling of each container. Using the CRISP-DM data mining methodology, it was possible to understand more about the business, collect, prepare, filter, and perform the optimized routes using the Google OR-Tools software, following the documentation of the Capacitated Vehicle Routing Problem. Environmental and financial factors were analyzed, with the projection of gains over time. The results showed a reduction of 8.5% in the distance spent and 17% in the time spent per route, which brings savings of more than 40 thousand Brazilian Reais and more than 1.5 tons of carbon dioxide dumped into the atmosphere per year for the company analyzed. There are still limitations such as the number of connected devices sending data and the number of analyzed routes, but information was provided on how to continue the project, which can be applied and customized according to the context.

Keywords: Routing; Selective Collection; Internet of Things; Google OR-Tools; CRISP-DM.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE GRÁFICOS

LISTA DE SIGLAS

ABRELPE	Associação Brasileira de Empresas de Limpeza Pública e Resíduos Especiais
API	*Application Programming Interface*
AS	*Ant Systems*
AS	*Simulated Annealing*
CDVRP	*Capacitated Distance Constrained Vehicle Routing Problem*
CRISP-DM	*CRoss-Industry Standard Process for Data Mining*
CO2	Dióxido de Carbono
CVRP	*Capacitated Vehicle Routing Problem*
CVRPTW	*Capacitated Vehicle Routing Problem with Time Windows*
DA	*Deterministic Annealing*
DF	Distrito Federal
EC	*Evolutionary Computation*
GA	*Genetic Algorithms*
GPS	*Global Positioning System*
GSM	*Global System for Mobile Communications*
HTTP	*Hypertext Transfer Protocol*
IaaS	*Infrastructure as a Service*
IBGE	Instituto Brasileiro de Geografia e Estatística
ID	*IDentity*
IoT	*Internet of Things* / Internet das Coisas
IISC	*IoT in Selective Collection*
IP	*Internet Protocol*
JSON	*JavaScript Object Notation*
LoRa	*Long Range*
NFC	*Near Field Communication*
NN	*Neural Networks*
PaaS	*Platform as a Service*
PBL	*Problem Based Learning*
PCV	Problema do Caixeiro Viajante
PNRS	Política Nacional de Resíduos Sólidos
PO	Pesquisa Operacional
RAM	*Random Access Memory*

REST	*Representational State Transfer*
RFID	*Radio Frequency Identification*
SaaS	*Software as a Service*
SBCs	*Single-Board Computers*
SDGs	*Sustainable Development Goals*
SOAP	*Simple Object Access Protocol*
SLU	Serviço de Limpeza Urbana do Distrito Federal
TCP/IP	*Transmission Control Protocol / Internet Protocol*
TS	*Tabu Search*
UIoT	*Universal Internet of Things*
UPnP	*Universal Plug and Play*
VNS	*Variable Neighborhood Search*
VRP	*Vehicle Routing Problem*
WCP	*Waste Collection Problem*
Wi-Fi	*Wireless Fidelity*
WoS	*Web of Science*
XML	*eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Um grande desafio do século 21 é gerenciar todos os resíduos produzidos pela humanidade todos os dias. A coleta seletiva surgiu como um mecanismo para separar os resíduos de acordo com suas características para auxiliar no manejo dos resíduos sólidos, conforme o crescimento da população urbana e da produção de lixo. No Brasil, 84,72% da população vive em área urbana (IBGE, 2015), sendo que as áreas urbanas do Brasil representam 0,63% do território nacional (Farias *et al.*, 2017), o que mostra a alta concentração populacional. Isso gera diversos problemas ambientais, como a produção de lixo, que de acordo com a Associação Brasileira de Empresas de Limpeza Pública e Resíduos Especiais (ABRELPE, 2020), a quantidade de resíduos coletados cresceu em todas as regiões do país e, em uma década, passou de cerca de 59 milhões de toneladas para 72,7 milhões de toneladas.

No Brasil, em 2010, foi instituída a Política Nacional de Resíduos Sólidos (PNRS), a Lei nº 12.305/10 que incentiva a coleta seletiva e o descarte correto dos resíduos. Segundo a PNRS, entende-se como resíduos sólidos todo:

> [...] material, substância, objeto ou bem descartado resultante de atividades humanas em sociedade, a cuja destinação final se procede, se propõe proceder ou se está obrigado a proceder, nos estados sólido ou semissólido, bem como gases contidos em recipientes e líquidos cujas particularidades tornem inviável o seu lançamento na rede pública de esgotos ou em corpos d'água, ou exijam para isso soluções técnica ou economicamente inviáveis em face da melhor tecnologia disponível. (BRASIL, Lei N° 12.305 de 02 de agosto de 2010 - Política Nacional de Resíduos Sólidos (PNRS)., p. 37).

Dessa forma, os resíduos sólidos podem ser plásticos, vidros, papéis, metais e a coleta seletiva desempenha um papel fundamental para separá-los e auxiliar no manejo correto de cada tipo de resíduo.

Um destaque entre os resíduos é o vidro, um material que é totalmente e infinitamente reciclável (Paynter e Jackson, 2016), o que categoriza um processo contínuo e infinito, trazendo a economia de água, energia elétrica, gás. Além disso, ele torna a atmosfera mais limpa, reduzindo normalmente 300 kg de CO2 para cada tonelada de vidro reciclada (Larsen, Merrild e Christensen, 2009).

A tecnologia é um fator facilitador na adoção de medidas sustentáveis urbanas, quebrando barreiras com a otimização dos processos. Já se tem visto algumas cidades que buscaram se tornar inteligentes a partir da utilização de sensores e do monitoramento de dados, como a cidade Santander, no Norte da Espanha, que já possui mais de 5000 sensores que monitoram o status dos pontos de descarga de lixo. Além disso, a cidade

conta com sensores relacionados a prevenção de vazamentos de água, promoção do turismo, gestão do tráfego, estacionamentos, iluminação e segurança (Díaz-Díaz, Muñoz e Pérez-González, 2017).

Pensando na coleta seletiva, a tecnologia e a utilização de sensores podem contribuir com o melhor entendimento do negócio a partir dos dados coletados, além da roteirização logística dos caminhões que realizam a coleta, economizando combustível, tempo e reduzindo o impacto ambiental gerado pela queima de combustíveis fósseis.

Foi nesse contexto que surgiu, na Dinamarca, um projeto para instalar em Brasília - DF sensores em contêineres de coleta seletiva de vidro para coletar dados e otimizar as rotas logísticas. O projeto buscou utilizar a comunicação via LoRa na coleta de lixo a partir da implementação de um sistema que mede o nível do lixo nos contêineres e envia para um servidor central. Foram realizados testes de performance e sugestões da implementação do projeto no contexto brasileiro, conforme apresentado em Hellmers *et al.* (2020).

No entanto, o projeto sofreu algumas alterações para ser aplicado na realidade brasileira, como o uso de outro protocolo de comunicação para envio dos dados e a utilização de painéis solares, aumentando a vida da bateria e utilizando uma energia renovável. Os dispositivos foram testados em uma empresa local de coleta seletiva de vidro e esse trabalho trata sobre a otimização das rotas dos caminhões para a coleta, comparando a diferença entre uma rota realizada no contexto atual e uma rota teórica a ser realizada a partir do uso do uso de sensores e Internet das Coisas.

Para a roteirização foi utilizado o *software* Google OR-Tools, de acordo com o *Capacitated Vehicle Routing Problem*, uma variação do Problema do Caixeiro Viajante que é um dos problemas mais relevantes no contexto de computação e matemática contemporânea (Aaronon, 2003). Para coletar, entender, tratar e preparar os dados, além da modelagem, avaliação e aplicação do modelo, foi utilizada a metodologia de mineração de dados *CRoss-Industry Standard Process for Data Mining*, seguindo as suas seis fases.

1.1 PROBLEMA DA PESQUISA

Segundo Hajghasem *et al.* (2016), os custos de transporte são uma grande parte dos custos logísticos, e aproximadamente 5 a 6% dos custos dos produtos vem dos custos de transporte. Dessa forma, a roteirização logística tem papel essencial no auxílio da redução de custos para uma empresa. De acordo com Mes, Schutten e Rivera (2014), uma

aplicação de um modelo de roteirização otimizada conseguiu reduzir em até 40% os custos de uma empresa de coleta de lixo.

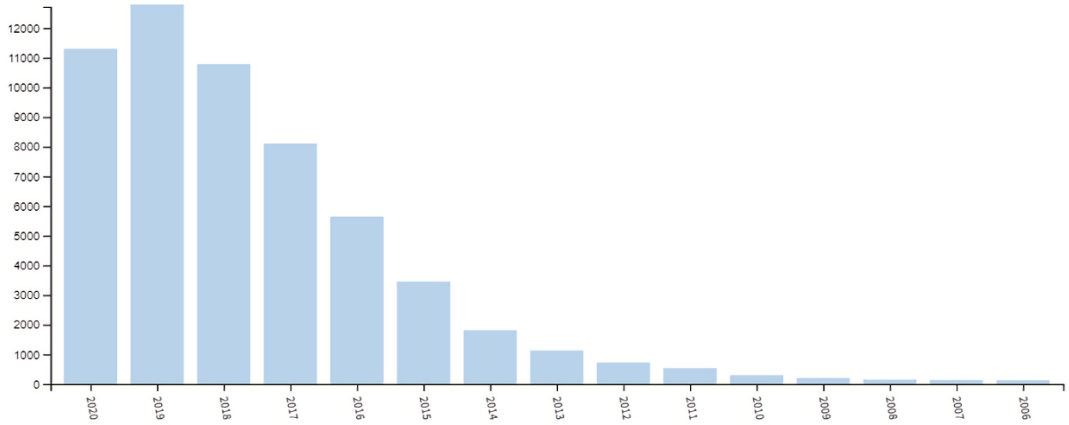
Sendo assim, a questão motivadora para esse trabalho é: Como uma solução de gerenciamento de coleta seletiva a partir do uso de internet das coisas e tecnologias de roteirização pode reduzir custos de uma empresa?

1.2 JUSTIFICATIVA

Este trabalho contribui diretamente com a preservação do meio ambiente, ao passo que a otimização de rotas logísticas traz uma menor emissão de combustíveis fósseis. Um algoritmo de roteirização de veículos apresentou uma redução de 7% das emissões de CO2 em um estudo de caso em que ele foi aplicado (Maden, Eglese e Black, 2010).

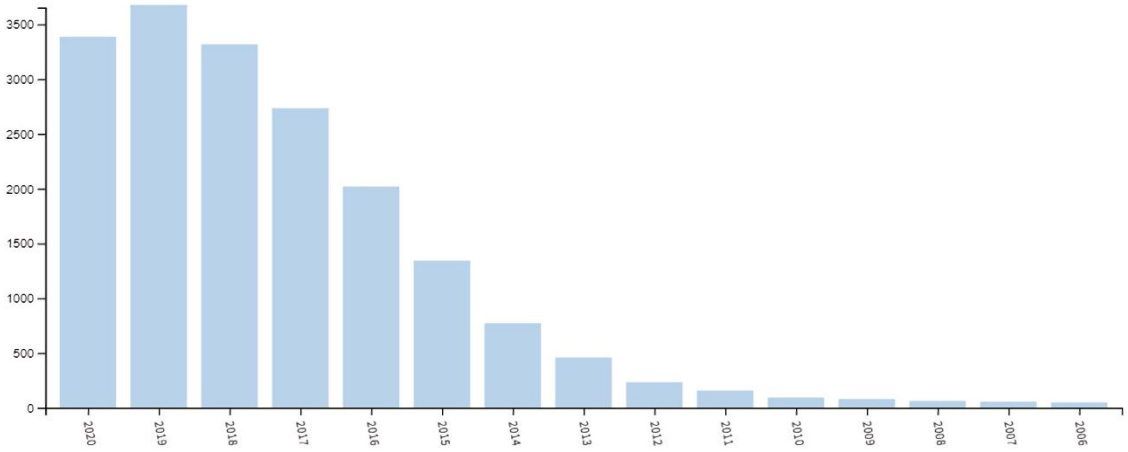
As publicações envolvendo IoT e cidades inteligentes crescem a cada ano, o que mostra o significativo interesse científico no tema, de acordo com o desenvolvimento tecnológico e uso das tecnologias na humanidade. Para observar esse crescimento, foram feitas buscas na plataforma de citações científicas Web of Science (WoS).

Figura 1 - Número de Publicações por ano de IoT no Web of Science de 2006 a 2020.



Fonte: Web of Science (2021).

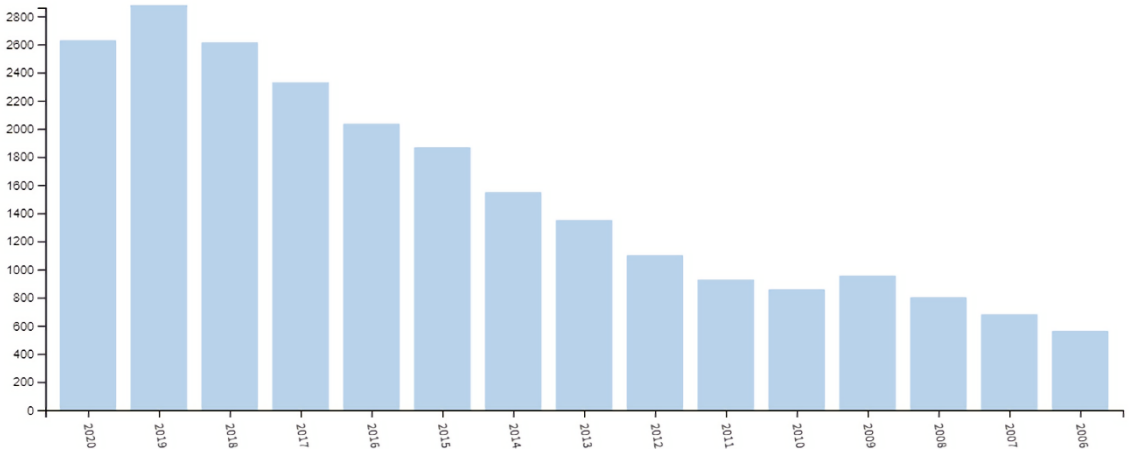
Figura 2 - Número de Publicações por ano de Cidades Inteligentes no Web of Science de 2006 a 2020.



Fonte: Web of Science (2021).

Além do crescimento das publicações sobre IoT e cidades inteligentes, pode-se observar o crescimento de publicações referentes a roteirização de veículos, portanto é um tema que acompanha o desenvolvimento da tecnologia e que pode ser aplicado em diversas situações.

Figura 3 - Número de Publicações por ano de Roteirização de Veículos no Web Of Science de 2006 a 2020.



Fonte: Web of Science (2021).

Este trabalho traz contribuições para diversas áreas da Engenharia de Produção, como o desenvolvimento de produtos, logística e Pesquisa Operacional (PO). Sobre o desenvolvimento de produtos, o trabalho trata sobre a tropicalização de um produto inicialmente pensado para a realidade da Dinamarca, o que gerou a necessidade de uma adaptação ao contexto brasileiro com etapas de desenvolvimento de produtos, como o levantamento de requisitos, prototipação e avaliação. Sobre logística e PO, o trabalho trata sobre o problema mais proeminente de otimização combinatória, o de encontrar a menor rota possível a partir de um conjunto de locais, o Problema do Caixeiro Viajante (PCV), que surgiu no século 19 e até hoje não possui uma solução totalmente satisfatória (Reinelt, 2003). A PO tem um grande papel no auxílio de redução de custos e na melhoria das operações, o que está alinhado com a redução de problemas ambientais criados pela produção, transporte, estoque e consumo de bens (Dekker, Bloemhof e Mallidis, 2012).

1.3 OBJETIVOS

1.3.1 Objetivo Geral

Aplicar um modelo de otimização de roteirização logística com o uso de Internet das Coisas em uma empresa que realiza a coleta seletiva, com o objetivo de reduzir o custo das rotas.

1.3.2 Objetivos Específicos

- Levantar, entender e tratar os dados-chave para a roteirização de uma empresa que realiza a coleta seletiva;
- Escolher modelo de otimização de rotas relevante e adequado para a implantação na empresa;
- Realizar a simulação de rotas otimizadas com o cálculo do custo; e
- Analisar a economia trazida pela implantação de um sistema de gerenciamento de rotas logísticas utilizando internet das coisas em uma empresa de coleta seletiva, nos aspectos financeiros e ambientais.

1.4 ESTRUTURA DO TRABALHO

O trabalho está estruturado da seguinte forma: o Capítulo 2 apresenta o Referencial Teórico que aborda Internet das Coisas, Roteirização e sobre a metodologia de Mineração de Dados utilizada. O Capítulo 3 apresenta as fases da metodologia aplicada. O Capítulo 4 aborda o estudo de caso e, o Capítulo 5 destaca os principais resultados a partir do modelo seguido pelo Capítulo 6 com as considerações finais, limitações e propostas de trabalhos futuros.

2 REFERENCIAL TEÓRICO

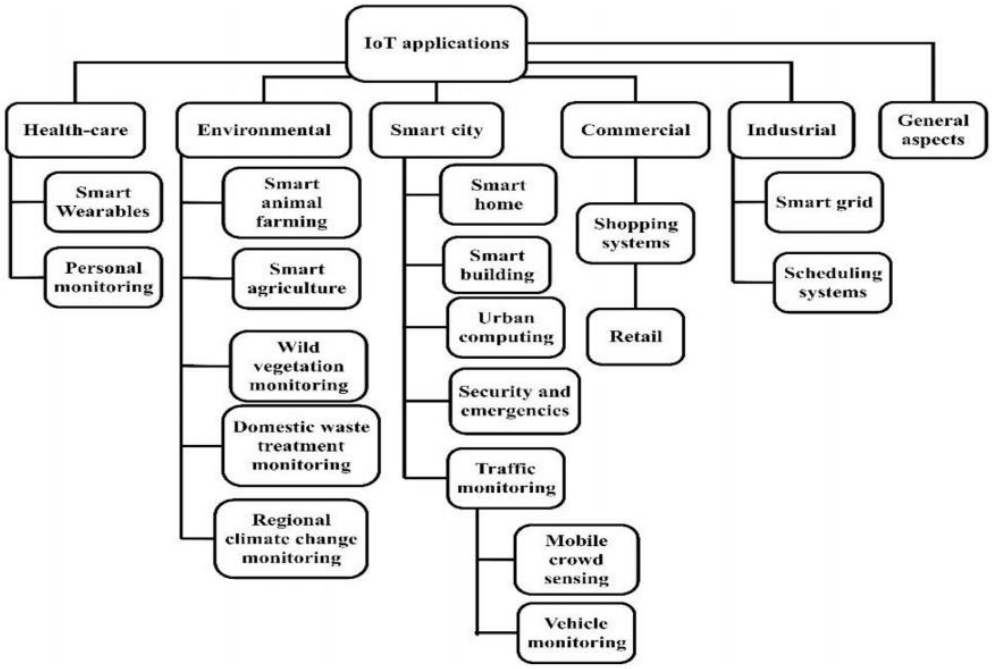
Para compreender melhor e fundamentar os conceitos utilizados no estudo de caso, este capítulo apresenta uma visão geral da arquitetura de um sistema de Internet das Coisas (IoT), além de trazer conceitos sobre roteirização logística e esclarecimentos sobre a metodologia de mineração de dados utilizada pelo trabalho.

2.1 INTERNET DAS COISAS

O termo Internet das Coisas, em inglês, *Internet of Things* (IoT), surgiu em uma apresentação feita por Ashton *et al.* (2009) tendo a maior relevância na época principalmente pela tecnologia do *Radio Frequency Identification* (RFID), acompanhada pelo amadurecimento e convergência das tecnologias com redes wireless e computação em nuvem (Greengard, 2015). Segundo Jia *et al.* (2012), a IoT se refere a objetos unicamente identificados e suas respectivas representações virtuais em uma estrutura que utiliza a internet, e se todos os objetos fossem equipados com *tags* RFID, eles poderiam ser identificados e catalogados por computadores.

A IoT é uma rede de redes, em que objetos, animais ou pessoas possuem identificadores únicos que podem transferir dados para uma rede sem precisar de interação homem-homem ou homem-máquina (Li, Da Xu e Zhao, 2015). Sabendo disso, pode se imaginar que as aplicações de IoT são vastas, desde o uso doméstico até o uso no trabalho e indústria:

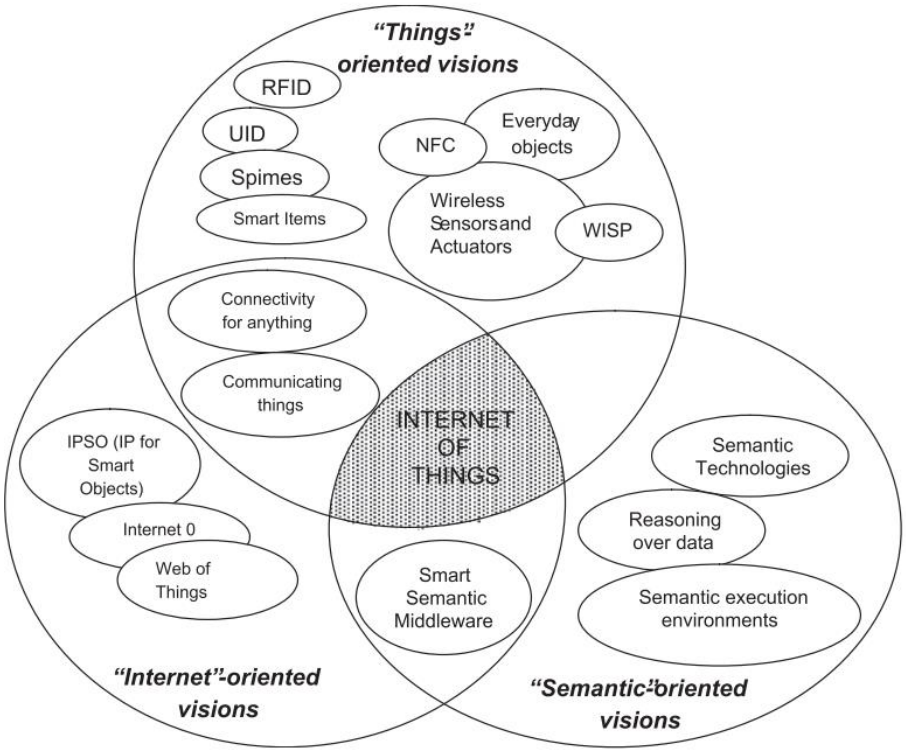
Figura 4 - Taxonomia das Aplicações IoT.



Fonte: Asghari, Rahmani e Javadi (2019).

Existem três perspectivas diferentes que formam o paradigma da IoT em seu ponto de convergência, a perspectiva orientada a coisas, a perspectiva orientada a semântica e a perspectiva orientada a internet (Atzori e Morabito, 2010):

Figura 5 - Paradigma da IoT.



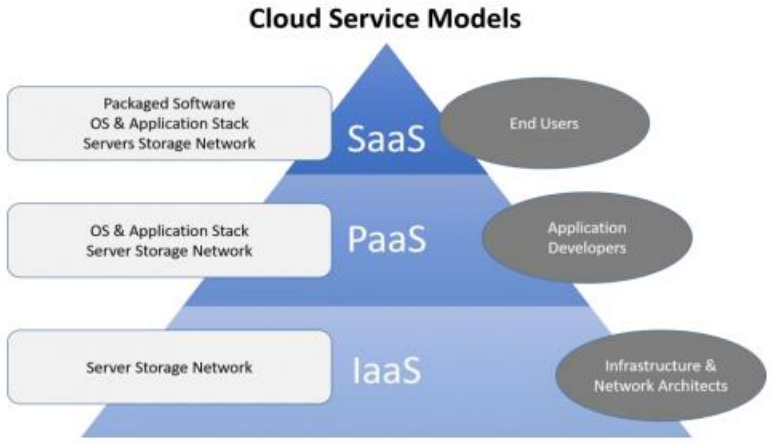
Fonte: Atzori e Morabito (2010).

A perspectiva orientada a coisas traz objetos do dia a dia com capacidade de comunicação e tem como objetivo monitorar objetos utilizando sensores e sistemas embarcados (Singh, Tripathi e Jara, 2014), tendo como ponto de convergência com a internet a comunicação e conexão desses dispositivos. Já a perspectiva orientada a internet tem o objetivo de tornar os objetos conectados, fornecendo características de protocolos de conexão, sendo o protocolo TCP/IP o mais comum, já que os sistemas embarcados normalmente possuem recursos limitados em termos de poder de processamento, memória e consumo de bateria (Chiang, 2006). A perspectiva orientada a semântica tem a ver com lidar, processar e limpar todos os dados gerados pelos dispositivos, que tendem a superar no futuro o número de páginas web que é aproximadamente 17 bilhões (Shahid e Aneja, 2017). Uma forma de lidar com esse alto número de dados é com o apoio da *Cloud Computing*.

2.1.1 *Cloud Computing*

A *Cloud Computing*, ou, Computação em Nuvem é um conceito chave para a IoT e leva a computação do computador local para toda a *internet*, sem o usuário precisar se preocupar com manutenção e gerenciamento dos recursos, o que permite que até mesmo um celular vire um grande *data center* (Aazam *et al*., 2014). Existem três tipos principais de serviços de *Cloud Computing*, que são *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS), e *Software as a service* (SaaS) (Ma e Zhang, 2012):

Figura 6 - Modelos de Serviços de *Cloud Computing.*



Fonte: Mohammed *et al.* (2021).

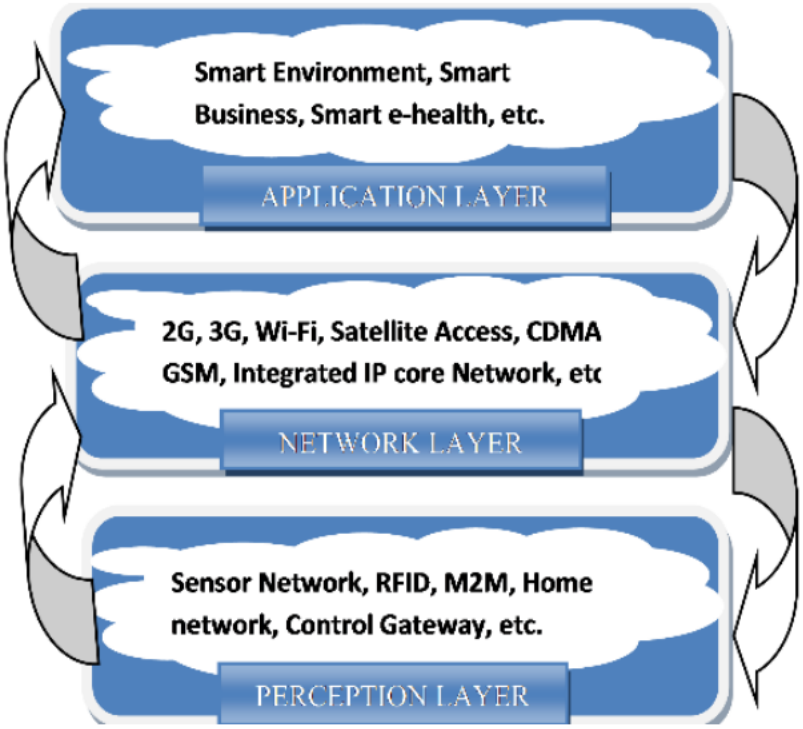
Nesse modelo, o serviço IaaS corresponde ao modelo mais básico em que o usuário não necessita lidar com a infraestrutura da nuvem, mas pode ter controle de sistemas operacionais, armazenamento, aplicações e de alguns componentes como *firewalls* (Miyachi, 2018). Já o serviço PaaS é mais focado nos desenvolvedores e corresponde ao modelo em que o usuário escreve suas aplicações utilizando linguagens e ferramentas suportadas por provedores e rodam elas na plataforma em nuvem (Ma e Zhang, 2012). O serviço SaaS é o que fica mais próximo do usuário final, permitindo o acesso aos usuários por meio de uma interface sem a necessidade de instalar, armazenar e manter aplicações (Aazam *et al.*, 2014).

A *Cloud Computing* tem capacidades virtualmente ilimitadas em termos de armazenamento e processamento, e tem a maioria dos problemas levantados pela IoT resolvidos, por isso são tecnologias complementares (Botta *et al.*, 2016)

2.1.2 Arquitetura Geral

Uma arquitetura típica e muito aceita de IoT é comumente dividida em três camadas, sendo a camada de percepção, rede e aplicações:

Figura 7 - Arquitetura Geral de IoT.



Fonte: Kumar e Patel (2014).

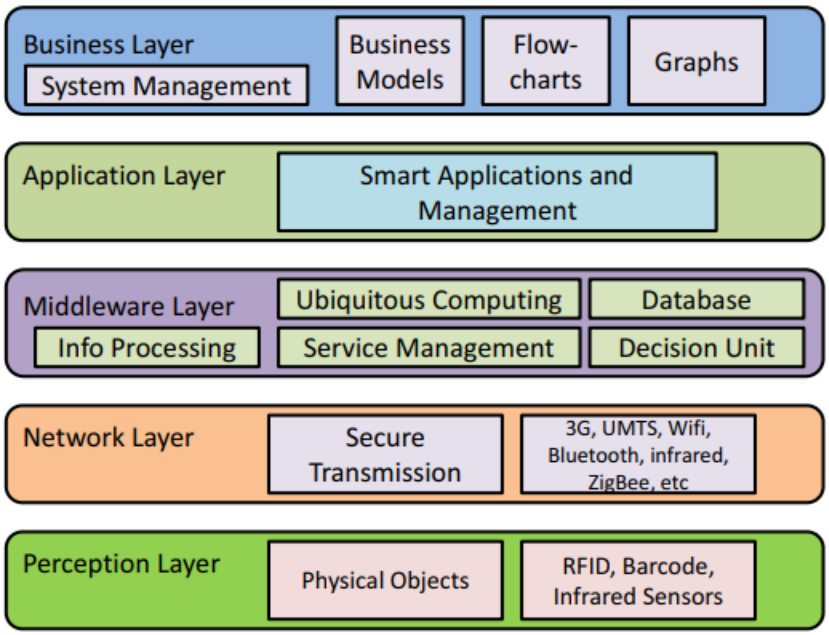
Nessa arquitetura, a primeira camada é a de percepção, em que os dispositivos e sensores dos mais variados tipos coletam informações e dados do mundo físico. Segundo Hassan (2018), as plataformas de desenvolvimento de hardware normalmente envolvem microcontroladores (Placas *Arduino*, *Particle Photon* e *Adafruit Feather Bluefruit*) e *Single-Board Computers* (SBCs) (*Raspberry Pi*, *BeagleBone Black* e C.H.I.P).

Já a camada de rede compreende o processamento inicial dos dados, transmissão, classificação e polimerização (Kumar e Patel, 2014). Segundo Talavera *et al.* (2017), essa camada inclui *gateways* que atuam como interface utilizando protocolos como o *Zigbee*, *Bluetooth*, *NFC*, *WiFi*, *LoRA* e *Sigfox*.

A camada de aplicações faz a interface entre os dados com o usuário, sendo comum o uso de *dashboards*, aplicações *web* e aplicativos para que o usuário ou gestor consiga acompanhar o *status* dos dispositivos. Essa camada combina a demanda social e industrial da IoT, de acordo com as características da tecnologia industrial e a necessidade da aplicação (Wu *et al.*, 2010).

Essa arquitetura de três camadas não descreve suficientemente as facetas de uma arquitetura de IoT, o que potencializou a criação de novos modelos, como o modelo que inclui uma camada de negócios e uma camada de *middleware* entre camada de rede e camada de aplicação:

Figura 8 - Nova Arquitetura Geral de IoT.



Fonte: Khan *et al.* (2012).

A camada do *middleware* é responsável pelo gerenciamento do serviço e está ligada ao banco de dados, processando informações e tomando decisões programadas a partir dos resultados desejados (Khan *et al.*, 2012). Um exemplo de *middleware* proposto por Ferreira *et al.* (2014) mostra uma solução escalável que lida com várias tecnologias como *UPnP*, *REST*, *Zigbee* e IP.

A camada de aplicações então se torna uma gestora das aplicações processadas pelo *middleware*, de acordo com a necessidade de uso. No modelo foi adicionada a camada de negócios, que traz informações chave para as atividades organizacionais, além de gerenciar a privacidade dos usuários e dar suporte a tomada de decisões (Wu *et al.*, 2010).

2.1.2.1 Sensores

Segundo Sehrawat e Gill (2019), os sensores têm um papel importante na camada de percepção ao detectar mudanças em condições físicas e podem ser simples ou complexos, sendo classificados de acordo com as suas especificações, métodos de conversão, tipo de material utilizado e outras propriedades:

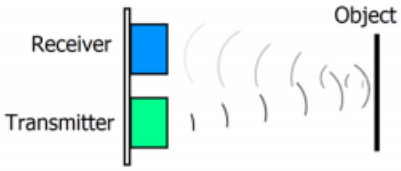
Figura 9 - Tipos de Sensores.

Fonte: Sehrawat e Gill (2019).

Como mostra a figura, os sensores de forma geral podem ser divididos em sensores de proximidade, posição, presença, movimento, velocidade, temperatura, pressão, umidade, gás, qualidade da água, infravermelho, giroscópio, óptica, química e outros de acordo com a necessidade do projeto. Especificamente falando dos sensores de proximidade, eles buscam detectar a posição de um objeto próximo sem contato físico, podendo ser indutivo, capacitivo, ultrassônico, fotoelétrico, magnético e de outros dependendo da aplicação.

Os sensores ultrassônicos são bastante utilizados para medir distâncias e se baseiam na medição do tempo de ondas ultrassônicas e têm a vantagem de não serem sensíveis a poeiras e poderem ser utilizados em ambientes menos controlados (Koval, Vaňuš e Bilík, 2016). Além disso, como mostra Zhmud *et al.* (2018), eles não são afetados pela luz solar ou pela cor dos objetos, como os infravermelhos e sua forma de funcionamento consiste em transmitir uma onda ultrassônica e perceber quanto tempo demora para ela voltar, convertendo para distância utilizando a velocidade do som:

Figura 10 - Funcionamento do Sensor Ultrassônico.



Fonte: Zhmud *et al.* (2018).

2.1.3 API

Segundo Sawant e Bacchelli (2015), uma API, ou *Application Programming Interfaces*, é um conjunto de funcionalidades providas por componentes de terceiros para que desenvolvedores de *software* possam utilizar. Atualmente, é muito raro encontrar projetos de *software* que não utilizem bibliotecas externas e suas APIs (Mileva, Dallmeier e Zeller, 2010), o que mostra a sua importância no mundo do desenvolvimento e naturalmente na IoT.

As APIs podem ser classificadas em privadas, de parceiros ou abertas, em que as privadas facilitam as informações dentro de organizações, as de parceiros fornecem suporte entre provedores e terceiros relacionados e as abertas fornecem acesso independente da relação com o criador (Ann e Iqbal, 2017). Pode-se imaginar que as APIs abertas possibilitaram uma revolução na utilização dos dados, já que desenvolvedores de todo o mundo podem realizar suas aplicações em conjunto utilizando APIs de empresas grandes e multinacionais.

Existem dois estilos principais de arquiteturas de APIs, o baseado no estilo REST e o baseado no protocolo SOAP (Adamczyk, 2011), sendo os acrônimos de *Representational state Transfer* e *Simple Object Access Protocol*, respectivamente. A API REST segue os padrões REST para criar serviços *web*, utilizando métodos para obter, adicionar, alterar e deletar informações (GET, POST, PUT, DELETE) com o protocolo HTTP e retornando respostas no formato JSON ou XML. A API SOAP também utiliza o protocolo HTTP, porém utiliza o formato XML e envia as mensagens em um formato de envelope contendo um *header* e um *body* (Soni e Ranga., 2019).

Para a IoT, dispositivos inteligentes falam entre si e transferem dados utilizando a internet, e as empresas fornecedoras de produtos de IoT provêm APIs para os seus clientes enviarem dados para *gateways* e servidores (Tatrous e Svennson, 2020), o que mostra a relevância dessas interfaces em várias camadas da arquitetura de IoT.

2.1.4 Internet das Coisas e Coleta de Resíduos

Existem diversas iniciativas no intuito da coleta seletiva e de resíduos em geral utilizando a tecnologia de IoT e de roteirização. Segundo Anagnostopoulos *et al.* (2018), a gestão de resíduos é tratada como um conjunto de serviços em cima de uma arquitetura de IoT em uma cidade inteligente, cobrindo o planejamento das rotas de coleta, o

transporte e a reciclagem. Por isso, já foram construídos diversos dispositivos para a coleta mais dinâmica, a partir da incorporação dos dados em sistemas inteligentes de roteirização.

Murugaanandam, Ganapathy e Balaji (2018) realizou uma comparação entre diversas arquiteturas utilizadas para tratar esse problema:

Figura 11 - Comparação de Arquiteturas de IoT na Coleta de Resíduos.

Name	Bin Measurement	Technology used	Object detection around the bin	Technology Used	Web UI	Alert Messages	Technology used	Scheduling
"IOT Based Intelligent Bin for Smart Cities"	Yes	Infra-red	No	No	No	Yes	GSM	No
"Efficient Garbage Disposal Management in Metropolitan"	Yes	laser, Photo diode	No	No	No	Yes	AD-HOC	Yes
"SMART DUSTBIN"	Yes	Ultrasonic	No	No	Yes	Yes	GSM	No
'smart garbage collection system in residential area"	Yes	Webcam, Weight sensor	No	No	LCD	No	GSM	No
"Waste Bin Monitoring System Using Integrated Technologies"	Yes	Ultrasonic	No	No	No	Yes	Zigbee GSM	No
"Smart Bin Implemented for Smart City"	Yes	Infrared	No	No	No	Yes	RFID	No
Basic Feature, "Solid waste Management project"	Yes	Webcam	Yes	Infra-red	No	Yes	GSM	No

Fonte: Murugaanandam, Ganapathy e Balaji (2018).

Como pode-se observar, existem diversas formas de implementação, variando a tecnologia utilizada para medir a ocupação dos contêineres, a forma de visualização de dados, a tecnologia utilizada para envio dos dados, a roteirização, entre outros. Conforme apresentado por Kang *et al.* (2020), os sistemas consistem em um sistema de medição, microcontroladores e uma forma de transmitir os dados, como *Global System for Mobile Communications* (GSM) ou Wi-Fi.

Um aspecto chave nas arquiteturas, é a redução do consumo energético, para que os aparelhos durem mais tempo, o que trouxe o surgimento de formas de redução. Algumas formas encontradas foram a utilização de painéis solares (Yusof *et al.*, 2018), a implementação do modo *sleep* para que o dispositivo não fique ligado o tempo todo (Samann, 2017) e a utilização de uma regra de envio apenas quando a ocupação passa do limite estabelecido (Kristanto *et al.*, 2016).

2.2 ROTEIRIZAÇÃO

Para falar sobre roteirização logística, uma série de termos devem ser explicados, iniciando pela Pesquisa Operacional, com uma explicação da teoria dos grafos, o surgimento do Problema do Caixeiro Viajante e suas variantes, a diferença de problemas P e NP e finalizando pelas heurísticas e algoritmos utilizados com fins de roteirização.

2.2.1 Pesquisa Operacional

Segundo Hillier e Lieberman (1995), a Pesquisa Operacional (PO) envolve pesquisa nas operações, em que a parte de pesquisa usa uma abordagem similar a utilizada em campos científicos. O processo começa observando com cuidado e formulando o problema, buscando todos os dados relevantes. O passo seguinte é construir um modelo científico (tipicamente matemático) que tenta abstrair a essência do problema real. Depois, assume-se que o modelo é uma representação suficientemente precisa das características essenciais da situação e que as conclusões obtidas pelo modelo são válidas para o problema real. Então, experimentos são conduzidos para testar as hipóteses, modificar se necessário e eventualmente verificar novas formas de hipóteses. Assim, para se ter sucesso a PO deve fornecer conclusões positivas e compreensíveis para o tomador de decisões.

Outra definição de PO vem da Operations Research Society Of America (Winston, 2004), que define como: “A pesquisa operacional busca decidir cientificamente como melhor desenhar e operar sistemas homem-máquina, normalmente sob condições que requerem a alocação de recursos escassos”.

Nota-se que a PO é ampla e abrange uma gama de organizações e aplicações, como no setor de transportes, já que podem se aplicar modelos para otimizar a logística e a roteirização em cidades, por exemplo. O conceito de logística de cidades pode ser definido como:

> [...] O processo para otimizar totalmente as atividades de logísticas e transporte de empresas privadas com o suporte de sistemas avançados de suporte em áreas urbanas considerando o ambiente de trânsito, o congestionamento, segurança e economia de energia com a estrutura de uma economia de mercado. (TANIGUCHI, Eiichi; THOMPSON, Russell G. (Ed.). City logistics: Mapping the future. CRC Press, 2014., p. 2).

2.2.1.1 Teoria dos Grafos

A teoria dos grafos surgiu em 1736 quando um matemático suíço buscou resolver um desafio proposto de fazer um trajeto passando por sete pontes, passando uma vez

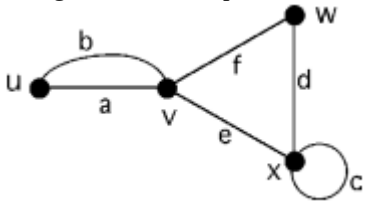
sobre cada, como mostra Sampaio (2015). Desde então ela evoluiu para uma série de problemas e aplicações, com uma ampla ligação com a PO e a ciência da computação.

Segundo Gross e Yellen (2003), qualquer objeto matemático envolvendo pontos e conexões entre eles pode ser chamado de grafo, existindo uma notação com 10 definições:

- **D1:** Um grafo $G = (V, E)$ consiste em dois elementos: V e E.
  - Os elementos de V são chamados de vértices (ou nós).
  - Os elementos de E são chamados de arestas.
  - Cada aresta tem um conjunto de um ou dois vértices associados, chamados de *endpoints*.
- **D2:** Se um vértice é um *endpoints* de uma aresta, então v é incidente em e e e é incidente em v.
- **D3:** Um vértice é adjacente ao outro se eles são encontrados por uma aresta.
- **D4:** Dois vértices adjacentes podem ser chamados de vizinhos.
- **D5:** Vértices adjacentes são dois vértices com um *endpoints* em comum.
- **D6:** Um vértice próprio é um vértice que junta dois vértices distintos.
- **D7:** Um vértice múltiplo é a coleção de dois ou mais vértices contendo *endpoints* idênticos.
- **D8:** Uma adjacência simples entre vértices ocorre quando existe exatamente uma aresta entre eles.
- **D9:** A multiplicidade de arestas entre um par de vértices é o número de arestas entre eles.
- **D10:** Um *self-loop* é uma aresta que encontra um *endpoints* dela mesma.

Sendo assim, um grafo pode ser visualizado para ficar mais clara a representação utilizada:

Figura 12 – Exemplo de Grafo



Fonte: Gross e Yellen (2003)

Este grafo possui os vértices $V = \{u, v, w, x\}$ e as arestas $E = \{a, b, c, d, e\ , f\}$, em que o conjunto $\{a, b\}$ é um vértice múltiplo com *endpoints* u e v e o vértice c é um *self-loop*.

2.2.1.2 O Problema do Caixeiro Viajante (PCV)

O PCV é um problema clássico de otimização combinatória em que se deve começar em uma cidade, passar por outras e retornar para a cidade com a melhor rota. Mais especificamente, segundo Laporte e Martello (1990), dado um grafo com valores associados a cada vértice, o PCV consiste em selecionar um circuito de otimização em que o comprimento não exceda a fronteira especificada. Em termos matemáticos, ele pode ser escrito como, dado uma matriz de distâncias de $n\ x\ n\ C = (cij)$, encontrar uma permutação cíclica π no conjunto $\{1,2, \dots, n\}$ em que ocorre a minimização da função:

Figura 13 – Função Minimizada do PCV.

$$c(\pi) = \sum_{i=1}^{n} c_{i\pi(i)}$$

Fonte: Burkard e Deĭneko (1998)

Esse problema evoluiu para uma série de problemas relacionados e a literatura pode ser encontrada em diversos trabalhos como os de Bodin (1981) e Golden e Assad (1986).

O grande desafio desse problema é quando ele inclui muitos nós e aumenta a complexidade de resolução, se tornando um problema de classe NP. Segundo Wigderson (2006), problemas de classe P são aqueles que uma computação eficiente será feita no tempo de execução do programa com entradas de dados de qualquer tamanho *n* vinculado a uma função polinomial em *n*.

No entanto, existem problemas que não se encaixam nessa classe, e segundo Wigderson (2006) existem centenas de problemas de classe NP na matemática, física e economia que não são resolvidos em décadas de esforço. O acrônimo NP significa tempo polinomial não-determinístico, em que o não-determinismo captura a habilidade de uma máquina não-determinística adivinhar uma solução e verificar ela deterministicamente.

Segundo Aaronson (2003), o problema P vs NP foi chamado de um dos problemas mais importantes na matemática contemporânea e na teoria da ciência da computação, e é uma das questões mais profundas já questionadas pelas quais nós deveríamos saber como reconhecer uma resposta.

2.2.1.3 As variações do PCV

Como já citado, o PCV possui diversas ramificações que merecem ser estudadas, a principal para este trabalho sendo O *Vehicle Routing Problem* (VRP). O VRP é um dos problemas mais estudados na PO (Costa, Contardo e Desaulniers, 2019) e foi introduzido por Dantzig e Ramser (1959) para resolver um problema de entrega de gasolina de um terminal para estações de serviço, buscando rotas para visitar todos os clientes uma única vez começando e terminando em um ponto e minimizando os custos de transporte.

Li *et al.* (2009) aponta que o VRP não é somente um dilema teórico na PO, já que não é suficiente obter as rotas de distribuição com algoritmos sob certas condições e por isso tem se usado a simulação computacional, uma atividade baseada em modelos que requerem modelos de simulação compatíveis com computadores.

O VRP ainda possui diversas variantes, com mudanças nos focos da solução e das funções objetivo, já que quando uma restrição é adicionada o problema muda, como mostra o quadro:

Quadro 1 - As variantes do VRP.

Problem	Objective	Focus	Comments
Traditional VRP	Minimize distance	Fleet	Toth (1981) proposed an overview of the traditional VRP, Cordeau (1997) provided an overview of the heuristic to solve this problem.
VRP with balance	Balance daily work	Driver	Levy and Bodin [1988] developed constraints to obtain good solutions with balanced daily work and Sniezek et al. (2006) extended this problem.
Period Vehicle Routing Problem (PVRP)	Account for time period constraints	Demand	Francis (2007) defined operational complexity as the difficulty in the PVRP
Inventory Routing Problem (IRP)	Ensure customer will not out of product	Demand	Demands in the IRP are stochastic based on monitoring and forecasts.
Consistent Vehicle Routing Problem (Con VRP)	Ensure customers are served with the same driver	Customer	Each driver has the same routes to visit every day. Wong (2008) proposed some practical issues.
Vehicle Routing Problem with Time Windows	Minimize travel cost within time windows	Customer	Specific time window constraints are considered in this problem.

Fonte: Li (2015).

Além disso, segundo Li (2015), existe a variação em que a capacidade de carga do veículo é incluída, sendo chamada de *Capacitated Vehicle Routing Problem* (CVRP) em que o objetivo é encontrar o caminho ótimo até que o veículo atinja a sua capacidade máxima de carga. O CVRP pode ser descrito como:

> [...]Um número de veículos idênticos com uma dada capacidade está localizado em um depósito central. Eles estão disponíveis para atender um conjunto de pedidos de consumidores (todas as entregas ou retiradas). Cada pedido tem uma localização e tamanho específico. Os custos de transporte entre todas as localizações são fornecidos. A meta é encontrar um conjunto de rotas com mínimo custo para os veículos de maneira que todos os consumidores sejam visitados somente uma vez e as capacidades dos veículos sejam respeitadas (HASLE, Geir; KLOSTER, Oddvar. Industrial vehicle routing. In: Geometric modelling, numerical simulation, and optimization. Springer, Berlin, Heidelberg, 2007. p. 397-435.).

De acordo com os trabalhos de Toth e Vigo (2002), pode-se analisar a fórmula matemática do CVRP, com sua função objetivo e restrições, como mostra a Figura 14:

Figura 14 – Função Objetivo e Restrições do CVRP.

$$Min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ijk} \quad (1)$$

subject to

$$\sum_{k \in K} \sum_{j \in \Delta+(i)} x_{ijk} = 1 \qquad \forall\, i \in N, \quad (2)$$

$$\sum_{j \in \Delta+(0)} x_{0jk} = 1 \qquad \forall\, k \in N, \quad (3)$$

$$\sum_{i \in \Delta+(j)} x_{ijk} - \sum_{i \in \Delta+(j)} x_{jik} = 0 \qquad \forall\, k \in N,\ j \in N, \quad (4)$$

$$\sum_{i \in \Delta-(n+1)} x_{i,n+1,k} = 1 \qquad \forall\, k \in K, \quad (5)$$

$$x_{ijk}(w_{ik} + s_i + t_{ij} - w_{jk}) \leq 0 \qquad \forall\, k \in K, (i,j) \in A, \quad (6)$$

$$a_i \sum_{j \in \Delta+(i)} x_{ijk} \leq w_{ik} \leq b_i \sum_{j \in \Delta+(i)} x_{ijk} \qquad \forall\, k \in K,\ i \in N, \quad (7)$$

$$E \leq w_{ik} \leq L \qquad \forall\, k \in K,\ i \in \{0, n+1\}, \quad (8)$$

$$\sum_{i \in N} d_i \sum_{j \in \Delta+(i)} x_{ijk} \leq C \qquad \forall\, k \in K, \quad (9)$$

$$x_{ijk} \geq 0 \qquad \forall\, k \in K,\ (i,j) \in A, \quad (10)$$

$$x_{ijk} \in \{0,1\} \qquad \forall\, k \in K,\ (i,j) \in A. \quad (11)$$

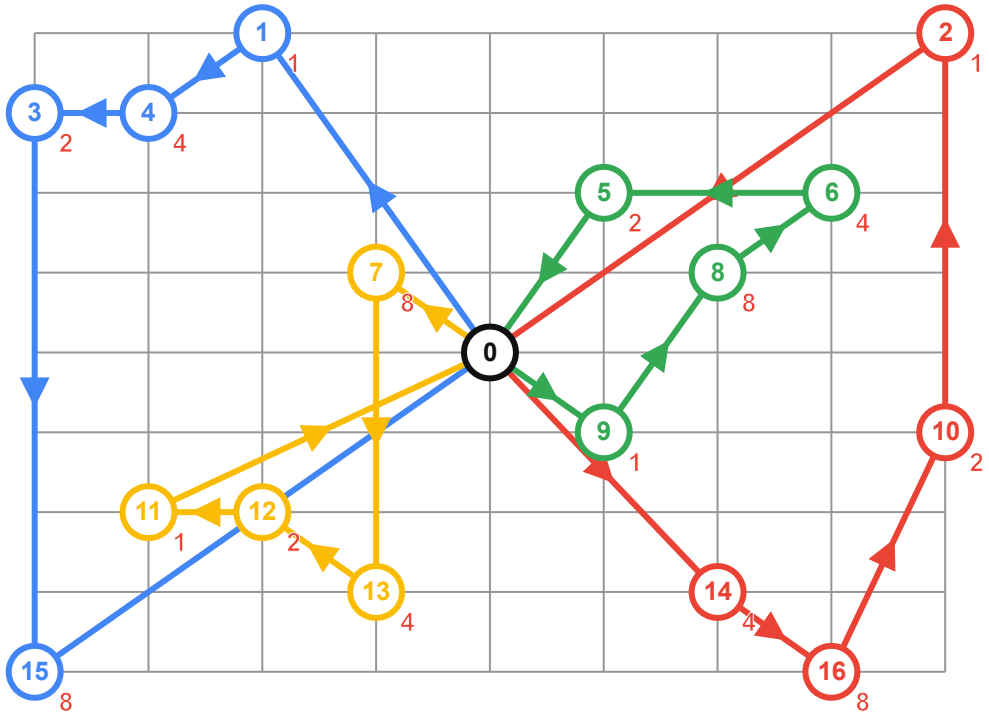
Fonte: Garzon Castro e Mondragón García (2020)

A seguir está descrita a função objetivo e as restrições:

> [...] A função objetivo 1 refere-se a minimizar o custo total expresso como unidades de distância na matriz de distância. A restrição 2 restringe a atribuição de cada cliente para exatamente uma rota de veículo. As restrições 3-5 caracterizam o fluxo no caminho a ser seguido pelo veículo k. As restrições 6-8 e 9 garantem a realização do cronograma no que diz respeito às questões de tempo e tamanho, correspondentemente. Observe que para um determinado k, a restrição 7 estebelece Wik = 0 sempre que o cliente i não for visitado pelo veículo k. Finalmente, a restrição 11 inflige condições binárias nas variáveis de fluxo. (GARZON CASTRO, Karina Andrea; MONDRAGÓN GARCÍA, Sebastián. A vehicle routing application for retail delivery with open source tools. 2020. p. 3.).

Um exemplo utilizado na documentação do *software* de otimização Google OR-Tools para o grafo do CVRP pode ser visualizado na Figura 15. Mais detalhes sobre o *software* estão descritos durante os seguintes capítulos do trabalho.

Figura 15 – Grafo Exemplo do CVRP.



Fonte: OR-Tools (2021)

Existe ainda o *Capacitated Vehicle Routing Problem with Time Windows* (CVRPTW), em que existe uma restrição temporal para a entrega e os consumidores podem ser atendidos em uma certa janela temporal e o *Capacitated Distance Constrained Vehicle Routing Problem* (CDVRP), em que existe a restrição do tamanho ou tempo máximo que não pode ser excedido para cada rota.

2.2.2 Algoritmos e Heurísticas de Roteirização

Como apresentado, o VRP, a não ser quando utilizado em pequenas instâncias, é um problema de classe NP, e por isso existem várias formas de tentar resolver o problema, divididas em algoritmos exatos, algoritmos de heurísticas clássicas e algoritmos de meta-heurísticas (Li, 2015). Os algoritmos exatos costumam ser baseados principalmente no método *Branch and Bound* (Toth e Vigo, 2002), além de programação dinâmica para algumas variações (Novoa e Storer, 2009). O método *Branch and Bound* é um procedimento que:

> [...] Consiste de uma enumeração sistemática de todas as soluções candidatas. Ele utiliza uma rotina de ramificação (branch) e uma rotina de verificação dos limitantes (bound). A primeira particiona o conjunto de soluções em subconjuntos menores e a segunda realiza a poda dos ramos por verificação dos limitantes. (OTA, Matheus J. et al. Algoritmo de Branch-Cut-and-Price para o Problema do Roteamento de Veículos Capacitados. Revista dos Trabalhos de Iniciação Científica da UNICAMP, n. 26, 2018. p. 5).

No entanto, os algoritmos exatos não conseguem resolver instâncias maiores do problema, como mostra Fukasawa *et al.* (2006), e por isso são utilizadas heurísticas que vêm evoluindo nos últimos quarenta anos, sendo divididas em heurísticas clássicas e meta-heurísticas (Cordeau *et al.*, 2002). Segundo Pearl (1984), a heurística é uma regra, estratégia, método ou artifício utilizado para melhorar a eficiência de um sistema que tenta descobrir soluções para problemas complexos. Dessa forma, as heurísticas oferecem soluções plausíveis e palpites inteligentes que indicam uma solução parcial para a solução do problema (Romanycia e Pelletier, 1985).

Segundo Laporte *et al.* (2000), as heurísticas clássicas performam uma exploração relativamente limitada do espaço de busca e geralmente produzem soluções de qualidade com um tempo modesto de computação. Já as meta-heurísticas não são nada mais do que melhorias procedurais sofisticadas e podem ser vistas como melhoramentos naturais das heurísticas clássicas.

2.2.2.1 Heurísticas clássicas

Entre as heurísticas clássicas mais conhecidas estão a de Clarke e Wright (1964), *Sweep, Petal, Greedy Algorithm, Nearest Neighbor* e *Mayer Method*.

Laporte *et al.* (2000) realizou uma comparação entre as principais heurísticas clássicas em relação ao tempo computacional:

Tabela 1 - Comparação Computacional de Heurísticas Clássicas.

n	Type[a]	Clarke and Wright[b]	Wark and Holt[c]	Sweep[d]		1-Petal algorithm[e]		2-Petal algorithm[f]		Fisher and Jaikumar[g]		Bramel and Simchi-Levi[h]		Best known solution value
50	C	578.56	524.6	531.90	0.12	531.90	0.10	524.61	0.76	524	9.3	524.6	68	524.61[i]
75	C	888.04	835.8	884.20	0.17	885.02	0.07	854.09	0.52	857	12.0	848.2	406	835.26[i]
100	C	878.70	830.7	846.34	1.18	836.34	0.32	830.40	3.84	833	17.7	832.9	890	826.14[i]
150	C	1128.24	1038.5	1075.38	2.53	1070.50	0.41	1054.62	5.93	1014	33.6	1088.6	2552	1028.42[i]
199	C	1386.84	1321.3	1396.05	3.60	1406.84	0.41	1354.23	6.21	1420	40.1	1461.2	4142	1291.45[i]
50	C, D	616.66	555.4	560.08	0.16	560.08	0.09	560.08	0.56	560	15.2	—	—	555.43[i]
75	C, D	974.79	911.8	965.51	0.19	968.89	0.07	922.75	0.43	916	20.6	—	—	909.63[i]
100	C, D	968.73	878.0	883.56	1.47	877.80	0.25	877.29	2.91	885	52.2	—	—	865.94[i]
150	C, D	1284.63	1176.5	1220.71	3.00	1220.20	0.26	1194.51	3.58	1230	121.3	—	—	1162.55[j]
199	C, D	1521.94	1418.3	1526.64	4.91	1515.95	0.35	1470.31	5.19	1518	136.6	—	—	1395.85[i]
120	C	1048.53	1043.4	1265.65	3.52	1252.84	0.61	1109.14	11.70	—	—	1051.5	1303	1042.11[i]
100	C	824.42	819.6	919.51	0.64	824.77	0.21	824.77	2.11	824	6.4	826.1	400	819.56[i]
120	C, D	1587.93	1548.3	1785.30	2.24	1773.69	0.26	1585.20	3.31	—	—	—	—	1541.14[j]
100	C, D	868.50	866.4	911.81	0.85	894.77	0.17	885.87	1.69	876	6.3	—	—	866.37[i]

[a] C: Capacity restrictions, D: distance restrictions.
[b] Parallel savings + 3-opt and best improvement, implemented by Laporte and Semet (2000). All computing times are negligible.
[c] Wark and Holt (1994). Best of five runs.
[d] Gillett and Miller (1974), implemented by Renaud et al. (1996a, 1996b).
[e] Foster and Ryan (1976), implemented by Renaud et al. (1996a, 1996b).
[f] Renaud et al. (1996a, 1996b). Computing times for the sweep, 1-petal and 2-petal heuristics are seconds on a Sun Sparcstation 2 (210.5 Mips, 4.2 Mflops), with 32 MB of RAM.
[g] Fisher and Jaikumar (1981). The rounding rule for distances is unspecified. Computing times are seconds on a DEC-10.
[h] Bramel and Simchi-Levi (1995). Computing times are seconds on an RS6000, Model 550.
[i] Taillard (1993).
[j] Rochat and Taillard (1995).

Fonte: Laporte *et al.* (2000).

As heurísticas apresentam resultados similares de acordo com o aumento de nós observados, com poucas variações significativas. Outro estudo comparativo entre heurísticas e algoritmos foi feito por Stopka, Jeřábek e Stopková (2020), avaliando a possibilidade do uso de cada uma delas para um problema de logística urbano e explicando os benefícios de cada uma:

Quadro 2 - Comparação do uso de Heurísticas e Algoritmos de Roteirização

Method name	Suitability for the application	Reason
Greedy algorithm	suitable	significant distance savings can be achieved even at a City logistics scale
Nearest Neighbor method	suitable	suitable for cases where only one supplier delivers to certain locations within urban territory
Least cost method	unsuitable	does not provide the optimal solution in lots of cases
Vogel approximation method	unsuitable	suitable for cases where multiple objects are operated by other objects over longer distances
Hungarian method	less suitable	the distribution task must be balanced
Little algorithm	less suitable	more suited to tasks with lower number of visiting places when traveling over longer distances
Clarke-Wright method	suitable	delivery route, when operating a large number of nodes, can be split into several partial routes
Mayer method	suitable	significant distance savings can be achieved even at a City logistics scale

Fonte: Stopka, Jeřábek e Stopková (2020)

O estudo observou que os métodos apropriados para o caso eram o *Greedy Algorith*, *Nearest Neighbor Algorithm*, Clarke e Wright e *Mayer Method*.

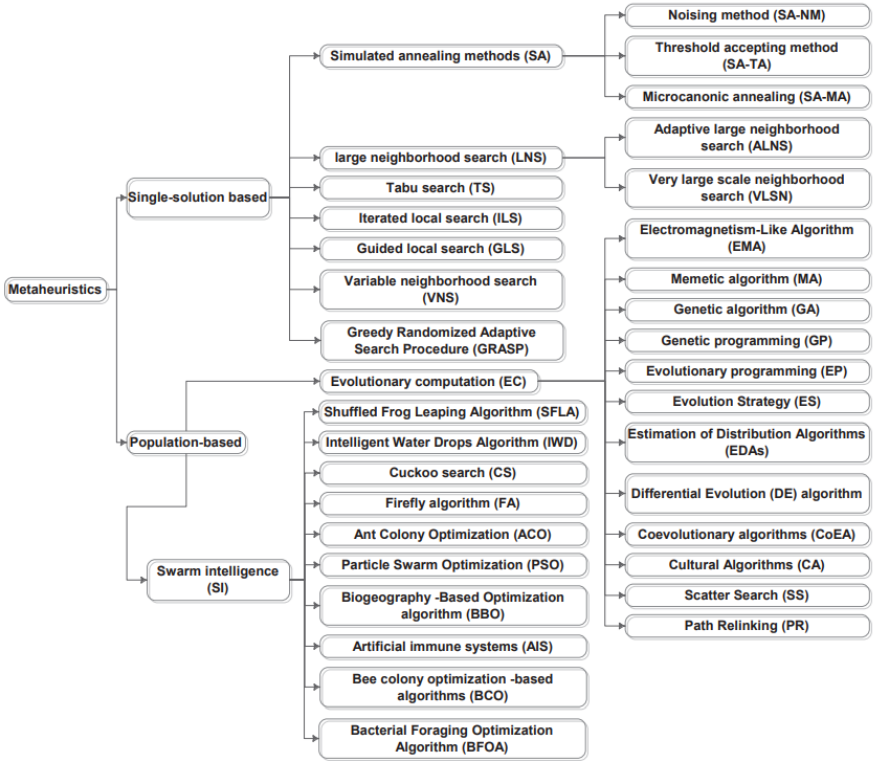
2.2.2.2 Meta-heurísticas

Segundo Gendreau e Potvin (2002), existem seis tipos principais de meta-heurísticas aplicadas para resolver o VRP: *Simulated Annealing* (SA), *Deterministic Annealing* (DA), *Tabu Search* (TS), *Genetic Algorithms* (GA), *Ant Systems* (AS) e Neural Networks (NN). Os três primeiros algoritmos começam em uma solução inicial e vão

movendo cada interação com os vizinhos até que uma condição de parada seja satisfeita. Os GA são estocásticos e o que os torna confiáveis é o processo de manter as melhores soluções a cada geração e usá-las para melhorar as próximas (Mirjalili, 2019). Os AS são inspirados no comportamento de espécies de formigas, em que elas depositam feromônios no chão para marcar os caminhos ótimos e o caminho mais curto é o que a maior parte das formigas tomou para levar a comida para o ninho (Dong, Guo e Tickle, 2012). As NN são inspiradas no mecanismo utilizado para processar informações pelo cérebro e pode ser caracterizada como uma função matemática não linear que transforma um conjunto de variáveis, de acordo com parâmetros que permitem a sua otimização (Bishop, 1944).

No entanto, por ser um problema muito comum, diversas soluções meta-heurísticas surgem a cada ano. Elshaer e Awad (2019) realizou uma revisão de literatura mapeando a maior parte das meta-heurísticas utilizadas:

Figura 16 - Comparação do uso de Heurísticas e Algoritmos de Roteirização.



Fonte: Elshaer e Awad (2019).

Foi percebido que os algoritmos são muitos diversos e há diversas formas de tentar resolver o problema do VRP, no entanto os resultados observados mostraram que as meta-heurísticas de soluções únicas mais utilizadas são as TS e VNS, as baseadas em população são a EC e GA (Elshaer e Awad, 2019).

2.2.2.3 *Softwares* de Otimização de Rotas

Pode-se observar que as soluções para o problema do VRP ainda estão surgindo e quanto mais o poder tecnológico das empresas e da sociedade cresce, mais a humanidade tem chance de resolver esse complexo problema. Muitas empresas têm interesse em resolver o problema para poupar os seus custos de logística e investem em soluções cada vez mais robustas, o que torna o mercado de sistemas de gerenciamento de frotas uma indústria que cresce rapidamente (Thong, Han e Rahman, 2007).

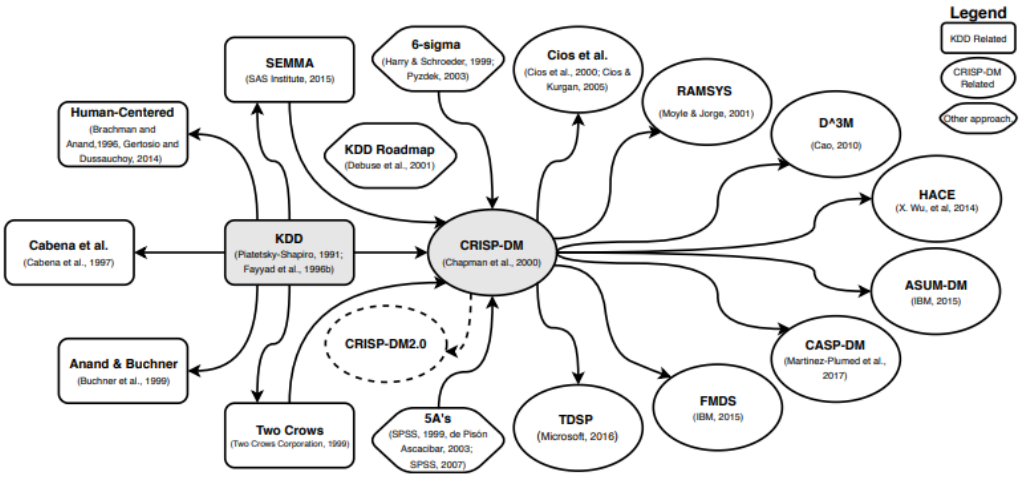
Baseado nos conceitos apresentados de API e de *Cloud Computing*, percebe-se que essa área tem grande relevância e o uso de serviços de empresas grandes se torna cada vez mais viável para a roteirização, como feito por Le e Pishva (2015) utilizando o serviço de API da Google para otimizar o sistema de distribuição para lojas de conveniência no Japão. Por meio do SaaS, o usuário não precisa se preocupar com a infraestrutura do sistema e com a evolução tecnológica das heurísticas e algoritmos, deixando isso na mão das áreas de pesquisa e desenvolvimento das grandes empresas provedoras de APIs de roteirização.

Segundo Bujel *et al.* (2018), existem alguns *frameworks* baseados nos algoritmos e meta-heurísticas mais eficientes do mercado, alguns sendo abertos e outros comerciais, sendo que alguns abertos são o Google Optimization Tools, ou Google OR-tools (Perron. 2011), SYMPHONY (Ralphs, Güzelsoy e Mahajan, 2010) e CVRPSEP (Lysgaard, Letchford e Eglese, 2004).

2.3 MODELO DE MINERAÇÃO DE DADOS

Segundo Wirth e Hipp (2000), a mineração de dados é um processo criativo que requer diversas habilidades e conhecimentos e necessita de uma abordagem para traduzir problemas de negócios em atividades de coleta de dados. Assim, em 1999, surge a primeira versão de um modelo chamado CRISP-DM (Chapman *et al.*, 2000), o que significa *CRoss-Industry Standard Process for Data Mining*. Esse modelo veio de uma evolução a partir da necessidade de se ter um método para projetos de exploração de dados e um panorama dessa evolução pode ser visto na Figura 17:

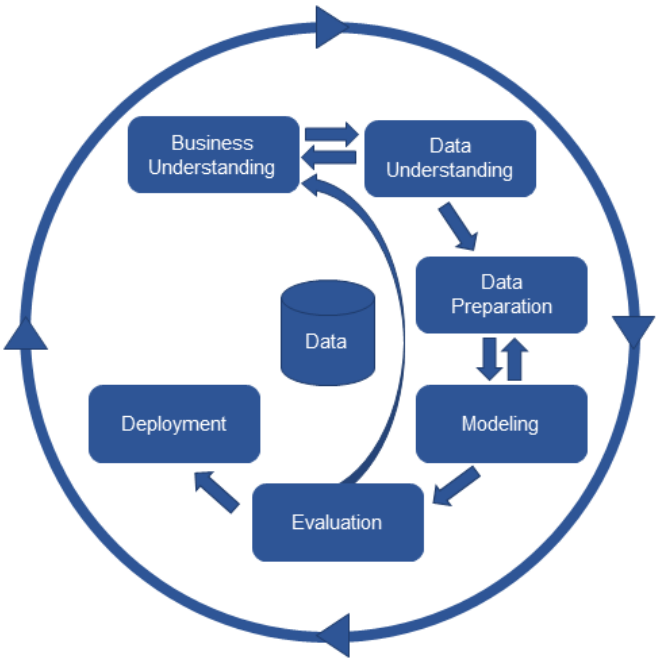
Figura 17 - Evolução dos Modelos de Mineração de Dados.



Fonte: Martínez-Plumed *et al.* (2019).

Como se pode observar, o CRISP-DM vem a partir de metodologias consolidadas e forma a base para uma série de propostas, sendo considerado a metodologia mais completa e mais utilizada para projetos de mineração de dados atendendo as necessidades até mesmo de projetos industriais (Martínez-Plumed *et al.*, 2019).

Figura 18 - Fases do CRISP-DM.



Fonte: Adaptado de Chapman *et al.* (2000).

Segundo Chapman *et al.* (2000) o modelo CRISP-DM possui seis fases:

- *Business Understanding*: essa fase inicial foca em entender os objetivos do projeto na perspectiva de negócios para entender quais dados devem ser coletados e trabalhados;
- *Data Understanding*: essa fase consiste em se familiarizar com os dados existentes sobre o negócio, identificando problemas de qualidade e formando os primeiros *insights;*

- *Data Preparation*: essa fase visa construir o conjunto de dados finais que será utilizado no modelo. As tarefas nessa fase costumam incluir a limpeza e seleção dos dados mais relevantes;
- *Modeling*: nessa fase são aplicadas várias técnicas de modelagem para calibrar os parâmetros e chegar no melhor modelo possível e é comum a necessidade de voltar para a fase de *Data Preparation* nesse processo;
- *Evaluation*: nessa fase o modelo já está calibrado da forma desejada com qualidade e é importante revisar se todos os dados-chave foram utilizados e é feita uma análise do modelo; e
- *Deployment*: a última fase consiste em aplicar o modelo de uma forma que ele possa ser visto em ação, comumente com modelos ao vivo em páginas web ou *dashboards*. Dependendo do caso, essa fase pode consistir na geração de um relatório, sendo flexível de acordo com as necessidades.

Sendo assim, o CRISP-DM é uma metodologia que pode ser utilizada para encontrar conhecimentos e trabalhar com dados até mesmo de pequenas e médias empresas, como dissertado por Bosnjak, Grljevic e Bosnjak (2009). Além disso, Meisel e Mattfeld (2010) apresenta que a mineração de dados e a PO têm uma grande sinergia em áreas como manufatura, logística e estoques, já que podem se beneficiar uma da outra, de um lado melhorando a eficiência da mineração de dados com os métodos de otimização e de outro aumentando a eficácia da PO com a mineração de dados.

3 METODOLOGIA

O estudo realizado foi um estudo de caso em uma empresa de coleta seletiva de vidro e seguiu o modelo da metodologia CRISP-DM descrito no capítulo 2.3, em todas as seis fases propostas por Chapman *et al.* (2000): *Business Understanding*, *Data Understanding*, *Data Preparation*, *Modeling*, *Evaluation* e *Deployment*. As fases foram seguidas após a identificação do tipo de problema do VRP e da escolha da ferramenta de roteirização.

O problema de roteirização de coleta seletiva foi identificado como CVRP, já que os caminhões levam as cargas coletadas nos contêineres até a ocupação máxima, além de que situações semelhantes de coleta de resíduos utilizaram a abordagem do CVRP, como o trabalho de Hannan *et al.* (2018) mostrou.

Para realizar a roteirização, o *software* Google OR-Tools foi escolhido, por possuir uma flexibilidade de algoritmos, ser *open-source*, ter uma maior facilidade para trabalhar com a integração de outras APIS da empresa Google como a de matriz de distâncias e tempos e a de trânsito, além de ter vencido a competição internacional de programação com restrições (Minizinc, 2020) por anos consecutivos. O trabalho de Surana (2019) executou diferentes experimentos variando as capacidades dos veículos, adicionando demandas diferentes nos nós com 12 diferentes estratégias e seis heurísticas para gerar 72 soluções diferentes utilizando o OR-Tools.

Surana (2019) também mostrou que aproximadamente 60% dos problemas têm uma solução produzida melhor pelo OR-Tools do que pelas soluções mais conhecidas no mercado. A documentação (OR-Tools, 2021) permite a utilização em diversas linguagens, no entanto a linguagem escolhida para a programação do modelo foi o *Python*, por maior contato do autor e simplicidade da programação e integração.

3.1 *BUSINESS UNDERSTANDING*

Para entender melhor sobre o negócio da empresa, foi realizada entrevista semiestruturada para entender o modelo de coleta seletiva atual, a variabilidade dos contêineres, dias de coleta e informações chave. Além disso, outros colaboradores da empresa foram entrevistados e indivíduos que possuem estabelecimentos próximos aos contêineres coletados, para conseguir informações de várias fontes sobre o modelo.

As principais informações coletadas nesse primeiro momento buscaram entender como é feita a coleta sem roteirização, a partir de valores quantitativos indicando:

- a periodicidade de coleta;
- a carga suportada por um caminhão;
- a carga média que um contêiner cheio costuma possuir;
- o número de caminhões no pátio da empresa;
- o consumo de combustível dos caminhões;
- o custo por hora por trabalhador, incluindo encargos; e
- o número de contêineres e localização.

Além disso, foram coletadas informações qualitativas, como a situação dos contêineres na hora da coleta, visando entender quanto o percentual de preenchimento varia de acordo com a região, a forma de roteirização que é feita atualmente, entender se há alguma forma de controle da localização atual dos caminhões via GPS e se são mapeados os custos com transporte especificamente. O questionário utilizado pode ser encontrado no Apêndice A.

3.2 *DATA UNDERSTANDING*

Essa fase visou traduzir e documentar os dados coletados na fase anterior, de forma a deixar mais fácil a preparação para rodar o modelo. Cada um dos dados coletados foi descrito:

- **Ocupação dos contêineres:**
  - Descrição: Dados variáveis medidos por sensores e enviados para o banco de dados.
  - Dados: Distância em centímetros, *id* do contêiner, data e hora de envio, nível da bateria.
- **Localização dos contêineres:**
  - Descrição: Local em que cada contêiner se encontra.
  - Dados: Latitude e longitude
- **Restrições da empresa:**
  - Descrição: Dados fixos obtidos a partir de entrevista com donos da empresa.
  - Dados: Número de caminhões disponíveis, carga suportada por caminhão, carga média por contêiner, custo por hora por trabalhador, o consumo de combustível dos caminhões.

3.3 *DATA PREPARATION*

Nessa fase os dados foram preparados para a utilização, prontos para serem inseridos no *software* utilizando o *Python* e o Google OR-Tools, seguindo 3 passos:

1 - Escolha dos contêineres que serão roteirizados, observando a quantidade total em kg dos contêineres mais cheios.

Dados de entrada: Capacidade máxima de um caminhão, ocupação atual dos contêineres e localização dos contêineres.

Processamento: Filtro dos contêineres que serão coletados.

Dados de saída: Planilha com localização, *id* e ocupação dos contêineres que serão coletados.

Figura 19 - Exemplo de Planilha dos Pontos Coletados.

ID	CLIENTE	coord
1	xxxx	-15.79542
2	xxxx	-15.68925
3	xxxx	-15.73811
4	xxxx	-15.71882
5	xxxx	583904,-47.88
6	xxxx	516948,-47.88
7	xxxx	500264,-47.88
8	xxxx	-15.75591
9	xxxx	383457,-47.89
10	xxxx	-15.74167
11	xxxx	580608,-47.89
12	xxxx	593062,-47.88
13	xxxx	-15.77443
14	xxxx	775305,-47.88
15	xxxx	88844,-47.893

Fonte: Autor (2021).

2 - Matriz distância e custo com cada um dos nós

Dados de entrada: Planilha com localização e *id* dos contêineres que serão coletados.

Processamento: Ler a planilha e conectar com a API de matriz de distâncias do Google.

Dados de saída: Matriz de distâncias e tempo por localizações.

Figura 20 - Exemplo de Planilha com Matriz de Distâncias e Tempo.

city_x	city_y	Distance in meter	duration in seconds
Bruges	Bruges	0	0
Bruges	Leuven	133079	5294
Bruges	London	285594	14031
Bruges	Stuttgart	631082	23694
Bruges	Ostend	29128	1726
Leuven	Bruges	126516	5151
Leuven	Leuven	0	0
Leuven	London	397871	17569

Fonte: Ewoud (2020).

3- Deixar as distâncias no formato para rodar no Google OR-Tools

Dados de entrada: Matriz de distâncias e tempo por localizações.

Processamento: Leitura da planilha e geração de lista de listas.

Dados de saída: Lista de listas com as distâncias para cada um dos nós selecionados.

Figura 21 - Exemplo de Formato de Lista de Listas

```
data['distance_matrix'] = [
    [
        0, 548, 776, 696, 582, 274, 502, 194, 308, 194, 536, 502, 388, 354,
        468, 776, 662
    ],
    [
        548, 0, 684, 308, 194, 502, 730, 354, 696, 742, 1084, 594, 480, 674,
        1016, 868, 1210
    ],
    [
        776, 684, 0, 992, 878, 502, 274, 810, 468, 742, 400, 1278, 1164,
        1130, 788, 1552, 754
    ],
```

Fonte: Autor (2021).

3.4 *MODELING*

Nessa fase, todos os dados preparados foram colocados no Google OR-Tools e foram feitas as remodelagens voltando na fase de *Data Preparation* até que o resultado fosse satisfatório, fornecendo a rota ótima para as localizações selecionadas. O código está no Apêndice B e pode ser visualizado em Luz (2021), disponibilizado no *site* GitHub com todo o histórico de construção e arquivos complementares que auxiliaram no processo. Para a construção, foram combinados trechos do código fornecido para o CVRP

pelos desenvolvedores do *solver* para a linguagem *Python* (OR-Tools, 2021) com códigos de consumo e preparação de dados utilizando a mesma linguagem feitos pelo autor e seguindo guias como o de Ewoud (2020). Mais detalhes da construção podem ser visualizados no Capítulo 5 e detalhes sobre o modelo de PO estão no Capítulo 2

3.5 *EVALUATION*

Nessa fase foi feita a avaliação de como o modelo rodou, observando os resultados das roteirizações e se os valores realmente faziam sentido. Essa fase foi descrita no Capítulo 5.

3.6 *DEPLOYMENT*

O modelo do CRISP-DM é adaptável a realidade dos trabalhos, e neste caso a fase de *deployment* está compreendida na análise dos resultados no capítulo 5 e nas futuras linhas de pesquisa no capítulo 6, em que se explicou como seguir com o modelo para que se tenha interação com o usuário e que seja possível uma visualização das rotas pelas partes interessadas.

4 ESTUDO DE CASO

Este capítulo traz a visão histórica do projeto que originou este trabalho e outros trabalhos relacionados, além do contexto em que ele está inserido e quais as contribuições trazidas.

4.1 O SURGIMENTO DO PROJETO

O surgimento do projeto deu-se a partir de uma parceria internacional entre a Universidade de Brasília no Brasil e a Universidade de Aalborg, na Dinamarca. A parceria ocorreu para desenvolver soluções tecnológicas que auxiliem a alcançar os *Sustainable Development Goals* (SDGs) adotados pela ONU em 2015 para que em 2030 o planeta esteja mais protegido e com desenvolvimento sustentável (UNDP, 2015). O desafio *The Global Students SDG Challenge* foi uma iniciativa chave para promover o surgimento do projeto, e a sincronia entre os estudantes se deu por conta da adoção das duas universidades do método *Problem Based Learning* (PBL), em que o aprendizado ocorre a partir de problemas.

Um dos projetos que foi fruto dessa parceria foi o *IoT in Selective Collection* (IISC), que impacta os objetivos 9,11 e 12 da Agenda 2030, sendo eles (UNDP, 2015):

- Objetivo 9: Construir infraestruturas resilientes, promover a industrialização inclusiva e sustentável e fomentar a inovação;
- Objetivo 11: Tornar as cidades e os assentamentos humanos inclusivos, seguros, resilientes e sustentáveis; e
- Objetivo 12: Assegurar padrões de produção e de consumo sustentáveis.

O intuito do IISC é auxiliar a coleta seletiva a partir do desenvolvimento de uma solução de *hardware* e *software* e o primeiro esforço relativo ao projeto foi realizado na Universidade de Aalborg (Hellmers *et al.*, 2020) no município de Aalborg na Dinamarca. A solução utiliza a tecnologia LoRa, contendo um dispositivo para realizar as medições dentro dos contêineres, um *gateway* e um servidor, além da realização de diversos testes:

Figura 22 - Solução Criada pela Universidade de Aalborg, contendo Sensor e *Gateway*.

Fonte: Hellmers *et al.* (2020).

No entanto, o sistema criado necessita de uma conexão WiFi para os *gateways* e algumas características que levaram a necessidade de criar outra solução mais adaptada à realidade brasileira.

4.2 ADAPTAÇÃO À REALIDADE BRASILEIRA

O projeto idealizado na Dinamarca em teoria seria instalado pelos próprios desenvolvedores em contêineres de uma empresa que realiza a coleta seletiva de vidro em Brasília, no intuito de testar o sistema e verificar sua aplicabilidade. No entanto, devido a complicações em relação a viagens internacionais com a pandemia do COVID-19, o projeto foi adaptado e a responsabilidade de instalar foi passada para o laboratório *Universal Internet of Things* (UIoT), que é um laboratório de pesquisa da Universidade de Brasília focado em criar uma arquitetura IoT universal, gratuita e dinâmica, além do desenvolvimento de soluções de *software* e *hardware*. Dessa forma, os membros do UIoT analisaram como o projeto poderia ser adaptado à realidade brasileira. O desenvolvimento de produtos de *hardware* durante a pandemia do COVID-19 ficou prejudicado, e o UIoT contou com uma adaptação no processo de desenvolvimento, que pode ser visualizada no trabalho de Patrão *et al.* (2020).

A primeira característica do sistema criado na Dinamarca é a necessidade de espalhar gateways pela cidade em pontos estratégicos com conexão *WiFi.* Na prática essa instalação é custosa e necessita de uma conexão boa de internet em vários pontos ou de uma rede pública, o que não é o caso de Brasília. Uma alternativa seria colocar em pontos parceiros da empresa usando a rede de cada ponto, no entanto isso demandaria um esforço estratégico com stakeholders e ficaria à mercê da boa vontade dos gestores de cada ponto.

Outro ponto levado em consideração foi a dificuldade em encontrar os componentes utilizados na concepção inicial, já que no Brasil os componentes eletrônicos presentes em maior escala para aplicações semelhantes eram outros e o laboratório UIoT já havia realizado a compra do período.

Em Brasília, as empresas não utilizam sensores para auxiliar na coleta seletiva, e segundo SLU (2020), existe um gasto de cerca de 8,3 milhões de reais mensais para o Serviço de Limpeza Urbana do Distrito Federal (SLU) realizar a coleta seletiva e a coleta de resíduos sólidos urbanos. Havendo uma iniciativa para investir em tecnologia na otimização da coleta, esse gasto pode diminuir e ser distribuído para outras causas e serviços. Na Figura 23, pode-se observar uma foto de como é feita a coleta, em que o caminhão do tipo Munck ergue o contêiner e abre a parte inferior para que os resíduos caiam na caçamba.

Figura 23 – Exemplo de Coleta.

Fonte: Autor (2021).

De forma a tentar manter ao máximo a identidade da concepção do sistema inicial, foi realizado um primeiro esforço em construir uma solução *LoRa* com os componentes presentes no Brasil e foi realizada a instalação em um contêiner da empresa.

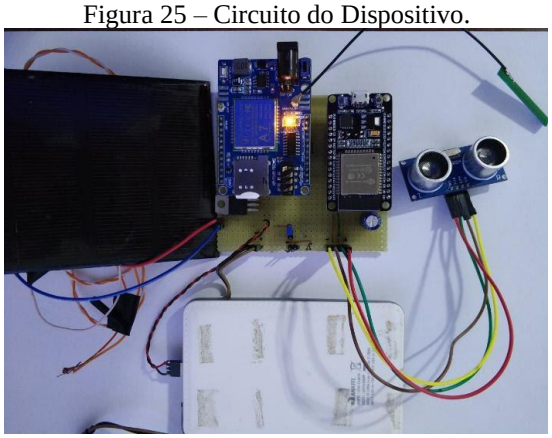
Figura 24 – Teste da Primeira Solução.

Fonte: Autor (2021).

No entanto, o alcance das antenas no teste foi muito baixo, o que levou a compra de antenas maiores e melhores, levando à melhoria do protótipo. Mesmo assim, os resultados alcançados não foram satisfatórios para a escalabilidade da solução, já que a empresa opera em distintos bairros do Distrito Federal e não possui uma distribuição densamente povoada, dificultando na hora de espalhar gateways em pontos estratégicos, o que é uma característica necessária para o uso da tecnologia *LoRa*.

Sendo assim, o grupo de desenvolvimento do UIoT optou por desenvolver uma arquitetura utilizando a tecnologia GSM, que elimina a necessidade de um gateway entre o dispositivo e o *middleware*, porém possui um maior consumo energético no envio dos dados. Dessa forma, a arquitetura também demandou a utilização de energia solar para promover a sustentabilidade.

Figura 25 – Circuito do Dispositivo.



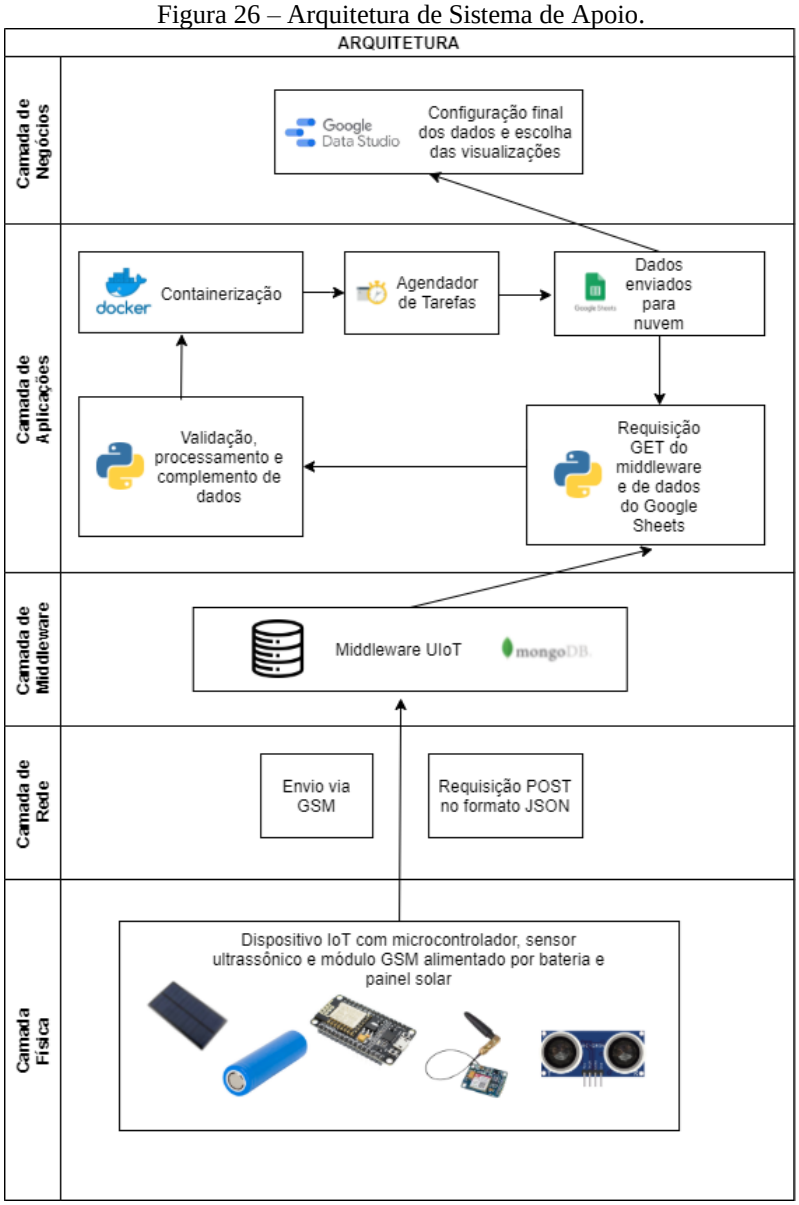
Fonte: Da Luz (2021, no prelo).

A arquitetura utilizando GSM mostrou sucesso em diversas etapas do desenvolvimento, sendo aprimorada constantemente para que o painel solar consiga suportar os envios, sendo o aspecto mais crítico o consumo de bateria. O trabalho de Monteiro *et al.* (2021, no prelo) detalha como foram feitos os testes de bateria para garantir a sustentabilidade do dispositivo.

Os dados coletados pelos sensores ultrassônicos da ocupação do preenchimento dos contêineres são enviados do dispositivo para o *middleware* aberto *UIoT* que possui o mesmo nome do grupo que o desenvolveu e possui uma arquitetura complexa para lidar com o envio de dados de dispositivos IoT utilizando o banco de dados não relacional *MongoDB*. O funcionamento dele pode ser visto em Do Prado *et al.* (2021) e Menezes e Costa (2019) que exibe os pilares utilizados para a construção, com ênfase na descentralização, processamento armazenamento e tomada de decisões.

O artigo de Da Luz *et al.* (2021, no prelo) demonstra como foi construído um sistema de tomada de decisões desde a camada física até a criação de um painel de acompanhamento, passando por todas as camadas de IoT, como mostra a Figura 26.

Figura 26 – Arquitetura de Sistema de Apoio.



Fonte: Da Luz *et al.* (2021, no prelo).

Tal arquitetura tem uma forte ligação com o projeto, possuindo uma atuação complementar para a implementação na empresa e fornecendo a infraestrutura necessária.

Foi criada uma rotina de tratamento e processamento de dados a partir do *middleware* que confere a qualidade necessária para que as informações que chegam ao gestor da empresa estejam filtradas corretamente e exibidas de forma intuitiva, como mostra a Figura 27.

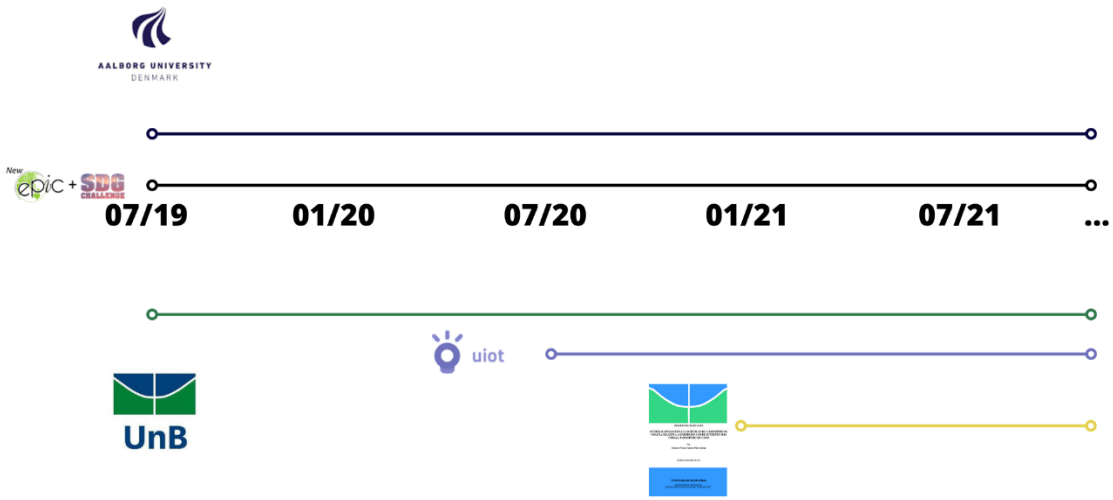
Figura 27 – *Dashboard* de Acompanhamento.

Fonte: Adaptado de Da Luz *et al.* (2021, no prelo).

4.3 CONTRIBUIÇÕES DESTE TRABALHO

Conforme apresentado nos objetivos, este trabalho trata sobre um modelo de roteirização, o que deve ser separado dos outros trabalhos realizados dentro do mesmo projeto, já que se trata de um desenvolvimento internacional e que passou por diversos grupos:

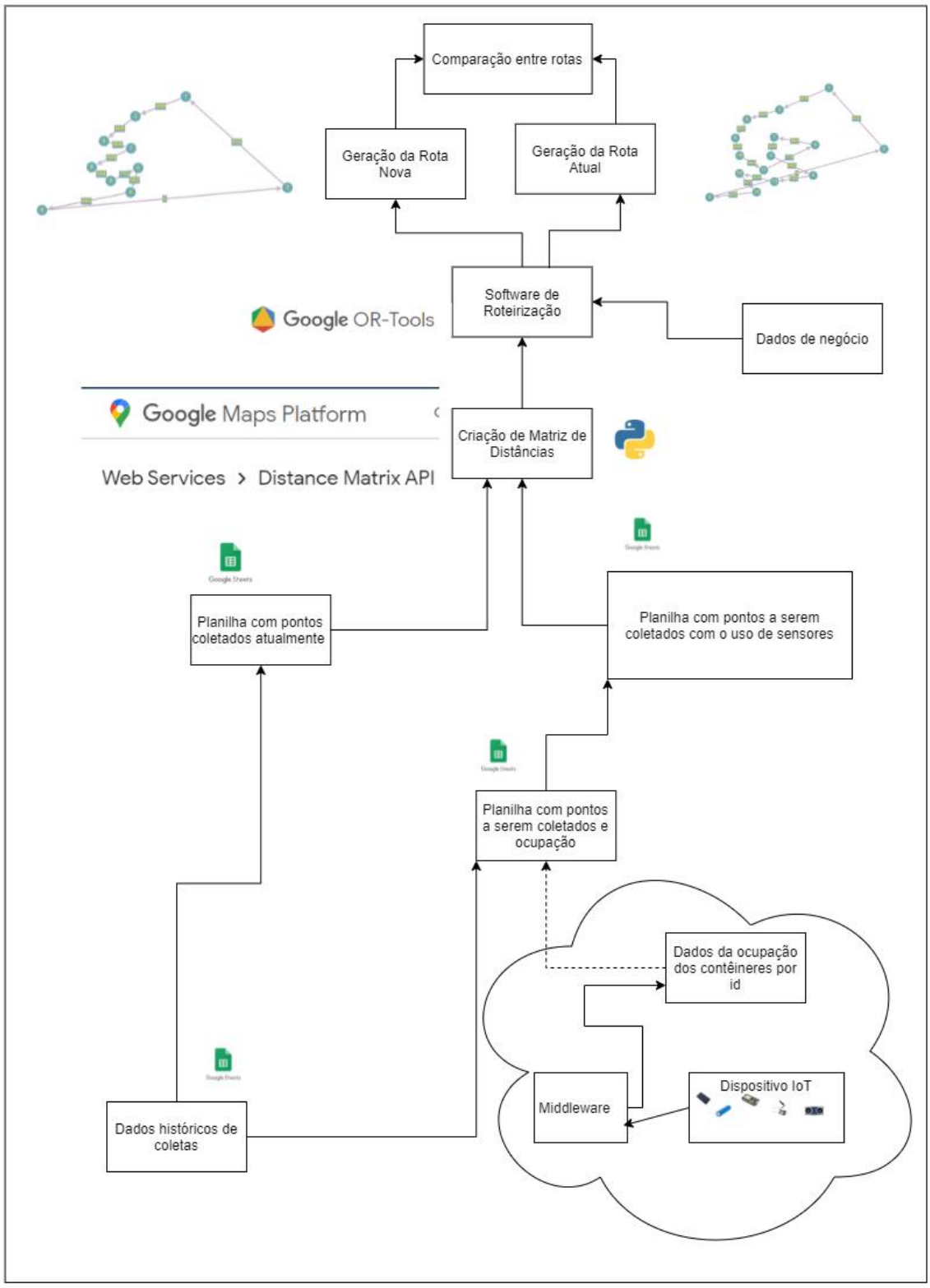
Figura 28 – Linha do Tempo do Projeto.



Fonte: Autor (2021).

O projeto iniciou em 2019 com a parceria entre a Universidade de Brasília e a Universidade de Aalborg trazida pelo desafio *The Global Students SDG Challenge*, e o laboratório UIoT começou a realizar a adaptação para a realidade brasileira no segundo semestre de 2020. O presente trabalho, em amarelo na Figura 28, trata da roteirização e da comparação entre rotas a partir do uso de sensores e as rotas realizadas historicamente pela empresa. A criação das rotas contou com o uso do *software* Google OR-Tools e a Figura 29 mostra como os dados foram trabalhados para que o *software* funcionasse como esperado, compreendendo as fases da metodologia.

Figura 29 – Etapas Utilizadas na Metodologia.



Fonte: Autor (2021).

O diagrama mostra as etapas utilizadas para que as rotas fossem criadas, exibindo a coleta, processamento e tratamento de dados em cada etapa. Primeiramente, existem os dados históricos das coletas contendo identificadores e coordenadas de cada contêiner. No cenário atual, a empresa descobre os pesos coletados durante a coleta, no entanto o

trabalho busca apresentar como seria a realização das rotas se houvesse a presença de sensores nos contêineres, por isso conta com a planilha com pontos a serem coletados.

Idealmente, os dados das pesagens viriam dos sensores da camada física e gerariam as capacidades de acordo com os identificadores dos contêineres, como mostra a parte dentro da nuvem da Figura 29. No entanto, o projeto ainda não está com a maturidade de possuir mais de 10 dispositivos enviando dados. Dessa forma, de acordo com a entrevista com o gestor da empresa e os dados históricos de rotas e pesos, foi feito um filtro de como seria a presença de um agente preditivo a partir do uso de sensores e os dados são divididos entre pontos coletados na rota atual e pontos a serem coletados na rota se houvessem sensores.

As duas planilhas com os pontos passam pelo mesmo processo de processamento utilizando a linguagem de programação *Python*, em que ocorre a leitura do arquivo, que é consumido pela API de matriz de distâncias da empresa Google e fornece uma entrada automatizada para o *software* Google OR-Tools, juntamente com os dados de negócio coletados via entrevista. Finalmente, as duas rotas são geradas permitindo a análise e comparação, que podem ser visualizadas no capítulo 5.

5 RESULTADOS E ANÁLISES

Este capítulo tem como objetivo apresentar a aplicação das etapas da metodologia CRISP-DM e está dividido na visão geral da coleta atual, visão da coleta com o uso de sensores, comparação de rotas e economia obtida. Cabe ressaltar que a empresa não foi identificada e os dados primários obtidos não foram divulgados para resguardar a privacidade e se adequar à Lei nº 13.709/18, ou Lei Geral de Proteção de Dados (Brasil, 2018), o que impactou na adaptação de figuras e exibições em todo o trabalho.

5.1 MODELAGEM E VISÃO GERAL DA COLETA ATUAL

A empresa possui 150 contêineres no total, e em geral, a coleta é realizada uma vez por semana de acordo com os grupos por setores da cidade. Para o trabalho, foi analisada uma dessas rotas realizadas tendo como parâmetro esse grupo para analisar o ganho trazido pela roteirização. Para realizar a coleta, existe um caminhão no pátio, que consegue carregar cerca de 9 toneladas, de acordo com a quantidade de vidro quebrada, o que depende da forma que o vidro é jogado e do tipo de garrafa.

Figura 30 – Pontos Coletados Atualmente.

ID	coo	PESAG	capacidade
1	-15.795	0	0%
2	-15.689	509	64%
3	-15.738	652	82%
4	-15.718	509	64%
5	3904,-47.8	665	83%
6	.6948,-47.8	502	63%
7	0264,-47.8	455	57%
8	-15.755	343	43%
9	3457,-47.8	565	71%
10	-15.741	559	70%
11	0608,-47.8	564	71%
12	3062,-47.8	715	89%
13	-15.774	473	59%
14	5305,-47.8	425	53%
15	8844,-47.89	432	54%

Fonte: Autor (2021).

Na rota atual, como mostra a Figura 30, a coleta passou por 14 pontos, sendo que cinco deles foram coletados com preenchimento menor que 60% da capacidade dos contêineres. Essa rota com as pesagens é processada e os dados são convertidos do formato .xlsx para um *dataframe*, uma estrutura de dados em forma bidimensional, de tamanho mutável, contendo dados tabulares e opcionalmente heterogêneos (Pandas, 2021). O *dataframe* é construído utilizando a linguagem de programação *Python* e é

gerada a matriz de distâncias entre os pontos utilizando a API do Google, em que o primeiro ponto é a sede da empresa de onde saem os caminhões.

A API fornece os dados em uma lista completa com as distâncias entre os pontos, porém para o Google OR-Tools processar os dados no código foi necessário converter a lista em uma lista de listas, utilizando a biblioteca *itertools* (Python, 2021) em que o tamanho de cada uma das listas é o número de contêineres. Esse artifício tornou o código mais escalável, já que a entrada de dados é moldada a partir da planilha inicial. A quantidade de carga de cada contêiner foi convertida em uma lista que é utilizada pelo Google OR-Tools, formando um dicionário juntamente com a capacidade dos veículos, o número de veículos, o ponto inicial da rota e a lista de listas como matriz de distâncias.

Os dicionários são coleções ordenadas (a partir da versão 3.7 do *Pyhton*) utilizadas para guardar dados com uma chave e um valor (W3Schools, 2021), em que a chave fornece a indexação e o valor contém os respectivos valores. Um exemplo de dicionário pode ser visto na Figura 31, sendo a entrada utilizada pelo *software*:

Figura 31 – Dicionário de Entrada do Google OR-Tools.

```
Entrada OR-Tools:  {'distance_matrix': [[0, 24941, 32528, 16127, 1581
1, 18543, 17671, 17984, 15054, 15182, 17446, 16946, 14251, 14241, 116
91], [25594, 0, 12225, 13927, 18457, 16067, 16917, 16816, 14300, 1428
5, 16693, 16192, 22131, 18220, 21914], [33474, 11980, 0, 22176, 26336
, 23947, 24797, 24696, 22180, 22165, 24573, 24072, 30011, 26100, 2979
4], [16908, 12700, 20287, 0, 8063, 5895, 6524, 6423, 3907, 3891, 6299
, 5798, 9508, 7827, 11520], [16739, 17594, 25181, 7694, 0, 4034, 2852
, 2996, 5235, 4943, 3284, 2784, 4241, 1915, 5766], [17048, 17456, 250
43, 7556, 2750, 0, 2736, 2880, 4375, 4085, 2436, 1935, 4412, 2549, 48
49], [18188, 16147, 23734, 6247, 2662, 1628, 0, 992, 3788, 3496, 1761
, 1260, 4950, 3365, 7080], [18274, 16233, 23820, 6333, 2986, 1714, 10
24, 0, 3874, 3376, 1847, 1346, 5036, 3451, 7145], [17073, 14184, 2177
1, 4284, 5321, 3267, 3782, 3681, 0, 580, 2795, 3057, 6165, 5085, 8779
], [17048, 14159, 21746, 4259, 5296, 3242, 3756, 3656, 652, 0, 2851,
3031, 6741, 5060, 8753], [16423, 16555, 24142, 6655, 3351, 1300, 1734
, 1878, 3159, 2836, 0, 513, 4204, 3115, 6808], [15910, 16042, 23628,
6141, 2837, 787, 1221, 1364, 2645, 2322, 583, 0, 3690, 2601, 6295], [
14156, 21911, 29498, 13096, 5440, 5596, 6030, 6174, 8278, 7955, 5392,
 4892, 0, 4713, 3827], [14508, 18097, 25684, 8197, 2067, 3366, 3355,
3499, 5433, 5143, 3494, 2993, 3714, 0, 4015], [12248, 21649, 29236, 1
2835, 4873, 6214, 6227, 6371, 8281, 6780, 5164, 4663, 4952, 2843, 0]]
, 'demands': [0, 509, 652, 509, 665, 502, 455, 343, 565, 559, 564, 71
5, 473, 425, 432], 'vehicle_capacities': [10000], 'num_vehicles': 1,
'depot': 0}   <class 'dict'>
```

Fonte: Autor (2021).

O Google OR-Tools para o CVRP possui no geral três funções: uma que retorna a distância entre dois nós, uma que retorna a demanda dos nós e outra que exibe a solução com as rotas. A modelagem do problema permite a inserção de pontos opcionais para customizar a rota, como a escolha da heurística utilizada na primeira solução e opções de busca local.

Figura 32 – Opções de Heurísticas da primeira solução.

Option	Description
AUTOMATIC	Lets the solver detect which strategy to use according to the model being solved.
PATH_CHEAPEST_ARC	Starting from a route "start" node, connect it to the node which produces the cheapest route segment, then extend the route by iterating on the last node added to the route.
PATH_MOST_CONSTRAINED_ARC	Similar to PATH_CHEAPEST_ARC, but arcs are evaluated with a comparison-based selector which will favor the most constrained arc first. To assign a selector to the routing model, use the method ArcIsMoreConstrainedThanArc().
EVALUATOR_STRATEGY	Similar to PATH_CHEAPEST_ARC, except that arc costs are evaluated using the function passed to SetFirstSolutionEvaluator().
SAVINGS	Savings algorithm (Clarke & Wright). Reference: Clarke, G. & Wright, J.W.: "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points", Operations Research, Vol. 12, 1964, pp. 568-581.
SWEEP	Sweep algorithm (Wren & Holliday). Reference: Anthony Wren & Alan Holliday: Computer Scheduling of Vehicles from One or More Depots to a Number of Delivery Points Operational Research Quarterly (1970-1977), Vol. 23, No. 3 (Sep., 1972), pp. 333-344.
CHRISTOFIDES	Christofides algorithm (actually a variant of the Christofides algorithm using a maximal matching instead of a maximum matching, which does not guarantee the 3/2 factor of the approximation on a metric travelling salesperson). Works on generic vehicle routing models by extending a route until no nodes can be inserted on it. Reference: Nicos Christofides, Worst-case analysis of a new heuristic for the travelling salesman problem, Report 388, Graduate School of Industrial Administration, CMU, 1976.
ALL_UNPERFORMED	Make all nodes inactive. Only finds a solution if nodes are optional (are element of a disjunction constraint with a finite penalty cost).
BEST_INSERTION	Iteratively build a solution by inserting the cheapest node at its cheapest position; the cost of insertion is based on the global cost function of the routing model. As of 2/2012, only works on models with optional nodes (with finite penalty costs).
PARALLEL_CHEAPEST_INSERTION	Iteratively build a solution by inserting the cheapest node at its cheapest position; the cost of insertion is based on the arc cost function. Is faster than BEST_INSERTION.
LOCAL_CHEAPEST_INSERTION	Iteratively build a solution by inserting each node at its cheapest position; the cost of insertion is based on the arc cost function. Differs from PARALLEL_CHEAPEST_INSERTION by the node selected for insertion; here nodes are considered in their order of creation. Is faster than PARALLEL_CHEAPEST_INSERTION.
GLOBAL_CHEAPEST_ARC	Iteratively connect two nodes which produce the cheapest route segment.
LOCAL_CHEAPEST_ARC	Select the first node with an unbound successor and connect it to the node which produces the cheapest route segment.
FIRST_UNBOUND_MIN_VALUE	Select the first node with an unbound successor and connect it to the first available node. This is equivalent to the CHOOSE_FIRST_UNBOUND strategy combined with ASSIGN_MIN_VALUE (cf. constraint_solver.h).

Fonte: OR-Tools (2021).

No trabalho foi escolhida a opção *AUTOMATIC*, em que o *solver* automaticamente seleciona a estratégia de solução, após a tentativa de selecionar heurísticas específicas. É possível utilizar técnicas avançadas de buscas locais, que permitem que o *solver* continue a busca após escapar de mínimos locais:

Figura 33 – Opções de Busca Local.

Option	Description
AUTOMATIC	Lets the solver select the metaheuristic.
GREEDY_DESCENT	Accepts improving (cost-reducing) local search neighbors until a local minimum is reached.
GUIDED_LOCAL_SEARCH	Uses guided local search to escape local minima (cf. http://en.wikipedia.org/wiki/Guided_Local_Search); this is generally the most efficient metaheuristic for vehicle routing.
SIMULATED_ANNEALING	Uses simulated annealing to escape local minima (cf. http://en.wikipedia.org/wiki/Simulated_annealing).
TABU_SEARCH	Uses tabu search to escape local minima (cf. http://en.wikipedia.org/wiki/Tabu_search).
OBJECTIVE_TABU_SEARCH	Uses tabu search on the objective value of solution to escape local minima

Fonte: OR-Tools (2021).

No trabalho foi escolhida a opção *GUIDED_LOCAL_SEARCH*, mencionada pela própria documentação como a meta-heurística mais eficiente em geral para a roteirização de veículos. A rota gerada pode ser visualizada na Figura 34.

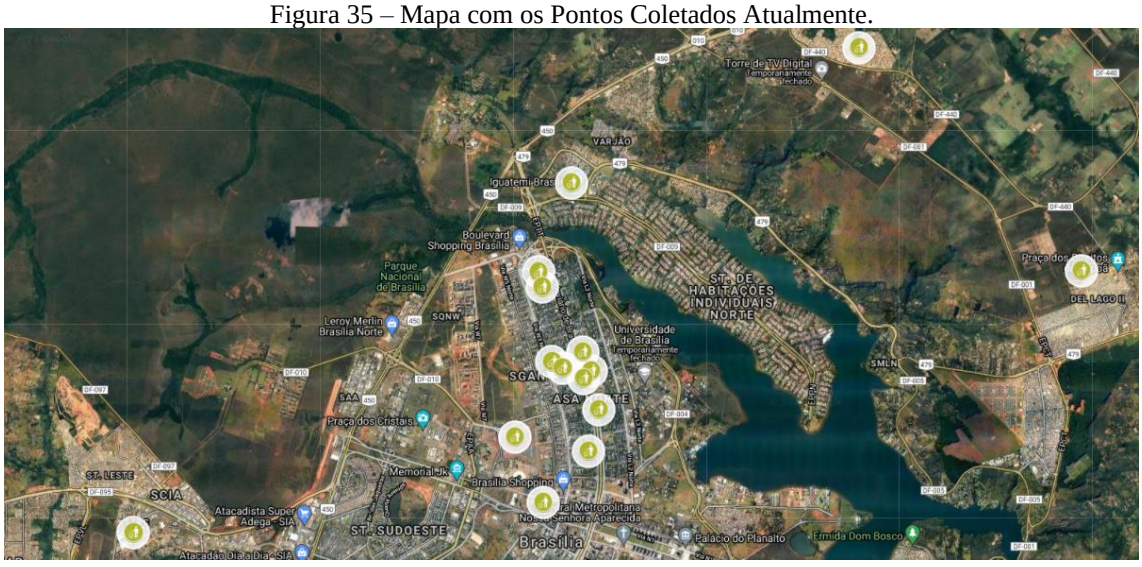
Figura 34 - Saídas do Código para a Rota Atual.

```
Objective: 94692
Route for vehicle 0:
 0 Load(0) ->  2 Load(652) ->  1 Load(11
61) ->  3 Load(1670) ->  8 Load(2235) ->
  9 Load(2794) ->  10 Load(3358) ->  11
Load(4073) ->  6 Load(4528) ->  7 Load(4
871) ->  5 Load(5373) ->  4 Load(6038) -
>  13 Load(6463) ->  12 Load(6936) ->  1
4 Load(7368) ->  0 Load(7368)
Distance of the route: 94692m
Load of the route: 7368
```

Fonte: Autor (2021).

Ao rodar o código, é exibido o objetivo da rota, que é a distância entre os pontos e a rota para o único veículo, de forma a exibir quanto foi coletado em cada ponto, qual a distância total e a carga total coletada. Os pontos coletados estão exibidos na Figura 35 e a rota está na Figura 36.

Figura 35 – Mapa com os Pontos Coletados Atualmente.



Fonte: Autor (2021)

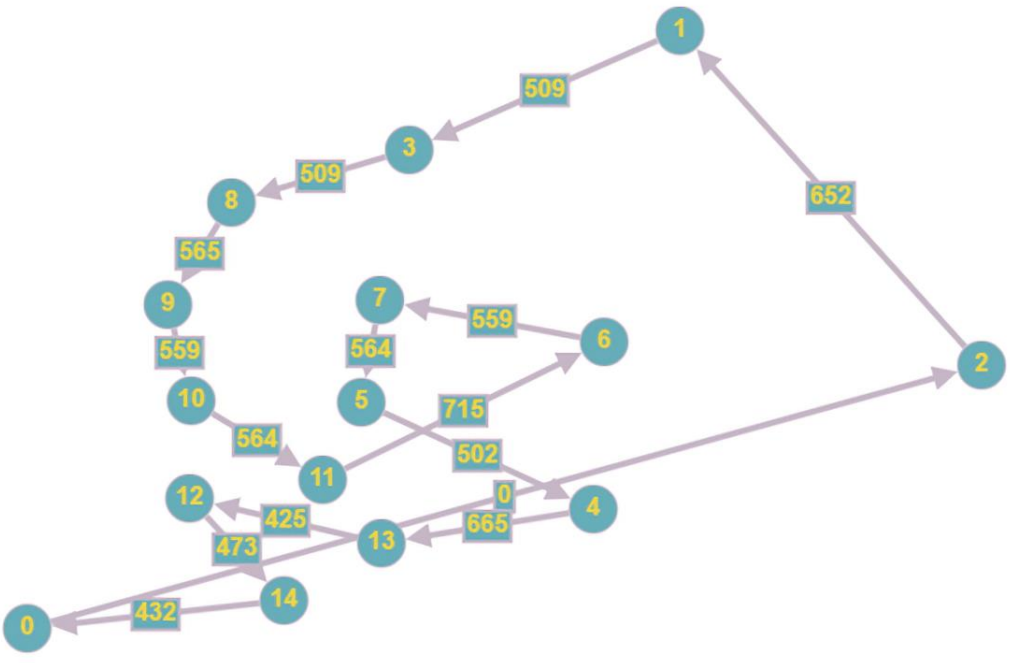
Figura 36 - Mapa com a Rota Realizada Atualmente.

Fonte: Autor (2021).

Nota-se que são coletados 14 pontos, em 2 horas e 21 minutos, percorrendo 94,7 quilômetros, em pontos bem espalhados pela cidade.

A visualização na forma de grafo é amplamente utilizada para realizar a análise de rotas, conforme exibido no Capítulo 2. Para a criação do grafo, foi utilizado o *software* de código aberto Graph Online (Graph Online, 2021), em que foram inseridos os pontos e pesos de acordo com as demandas coletadas, além da conexão entre os vértices.

Figura 37 - Grafo com a Rota Realizada Atualmente.



Fonte: Autor (2021).

O grafo confirma a coleta de vários pontos com a demanda menor que a necessária e a alta concentração de pontos coletados, podendo ser simplificado.

5.2 VISÃO DA COLETA SENSORIZADA

Para analisar qual seria a contribuição da utilização de IoT na coleta seletiva, considerou-se a rota nova como sendo a mesma rota atual, sem passar pelos pontos em que a coleta não seria necessária. A Figura 38 mostra os pontos contidos na rota proposta e a Figura 39 mostra as saídas do código.

Figura 38 – Pontos Contidos na Rota Proposta.

ID	COO	PESAG	capacidade
1	-15.7954	0	0%
2	-15.6892	509	64%
3	-15.7381	652	82%
4	-15.7188	509	64%
5	33904,-47.88	665	83%
6	16948,-47.88	502	63%
7	33457,-47.89	565	71%
8	-15.7416	559	70%
9	30608,-47.89	564	71%
10	93062,-47.88	715	89%

Fonte: Autor (2021).

A rota é uma versão reduzida da exibida na Figura 30, simulando como seria se somente os contêineres necessários fossem coletados a partir do uso de sensores fornecendo as pesagens.

Figura 39 - Saídas do Código da Rota Proposta.

```
Objective: 86562
Route for vehicle 0:
 0 Load(0) ->  2 Load(652) ->  1 Load(11
61) ->  3 Load(1670) ->  6 Load(2235) ->
  7 Load(2794) ->  8 Load(3358) ->  9 Lo
ad(4073) ->  5 Load(4575) ->  4 Load(524
0) ->  0 Load(5240)
Distance of the route: 86562m
Load of the route: 5240

Total distance of all routes: 86562m
Total load of all routes: 5240
```

Fonte: Autor (2021).

Os dados passaram pelo mesmo processo de tratamento e construção da outra rota, alterando somente a quantidade de pontos, já que a entrada de dados é adaptável de acordo com o número de contêineres. A comparação entre as rotas pode ser visualizada nas próximas seções. Os pontos que seriam coletados estão exibidos na Figura 40 e a rota está na Figura 41.

Figura 40 - Mapa com os Pontos Coletados da Rota Proposta.

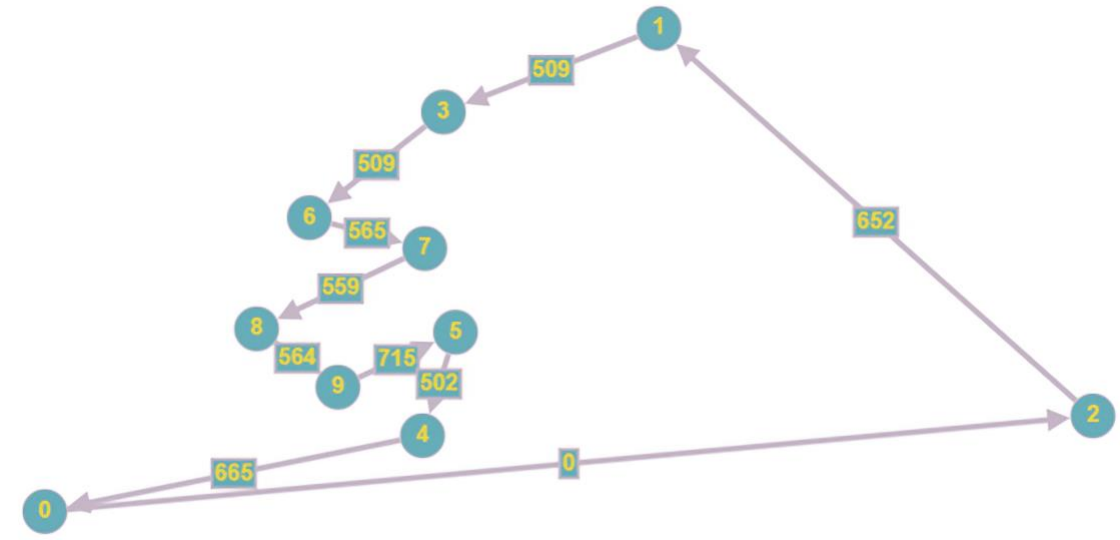
Fonte: Autor (2021).

Figura 41 - Mapa com a Rota Proposta.

Fonte: Autor (2021).

Nota-se que seriam coletados 9 pontos, em 1 hora e 57 minutos, percorrendo 86,6 quilômetros. A visualização da rota também está disponível na forma de grafo.

Figura 42 – Grafo com a Rota Proposta.



Fonte: Autor (2021).

O grafo mostra de forma geral como a coleta ficou reduzida e mais inteligente com o menor número de nós, coletando somente os pontos críticos da ocasião.

5.3 COMPARAÇÃO DAS ROTAS

A comparação dos pontos visitados entre as duas rotas está na Figura 43, em que os pontos em verde escuro simbolizam a coleta com o auxílio de sensores e os pontos em verde claro são os que também são coletados na rota atual.

Figura 43 – Comparação dos Pontos Visitados.

Fonte: Autor (2021).

Pode se notar que a rota não alterou tanto nas longas distâncias, e sim nas paradas na região do Plano Piloto de Brasília, indicadas pelos pontos verde claros. O resultado do tempo e distância gastos são exibidos na Figura 44.

Figura 44 – Gastos e Economias de Tempo e Distâncias por Rota.

	rota antes
km	94.7
tempo (minutos)	141
tempo (horas)	2h21
	rota depois
km	86.6
tempo (minutos)	117
tempo (horas)	1h57
	economia rota
km	8.1
tempo	24

Fonte: Autor (2021).

Percebe-se que a nova rota possui 8,1 quilômetros a menos e é feita em 24 minutos a menos, sem considerar o tempo que seria gasto para realizar a coleta em cada um dos pontos não visitados. Essa economia correspondeu a 8,5% de distância e 17% de tempo. Durante o processo de construção e avaliação do código foram feitas 8320 solicitações para a API de matriz de distâncias da empresa Google, com nenhum erro e latência média de 19 milissegundos, o que é exibido no painel do Google Cloud Platform, uma solução de *Cloud Computing*:

Figura 45 – Utilização da API de Matriz de Distâncias.

API ↑	Solicitações	Erros	Latência média (ms)
Distance Matrix API	8.320	-	19

Fonte: Google Cloud Platform (2021).

O computador utilizado para a execução do código possui um processador Core i7 8565U, da marca Intel com 8 *gigabytes* de memória *Random Access Memory* (RAM). A execução do código da rota atual durou 30 segundos e o da rota nova durou 14 segundos.

5.4 ECONOMIA OBTIDA

Para calcular a economia obtida, primeiramente fez-se o cálculo do custo por quilômetro do caminhão do tipo Munck, o utilizado para realizar a coleta. Após esse cálculo, foi considerada a economia da rota de amostra e foi possível obter a economia por semana, mês e ano a partir do número de rotas realizadas por semana, como mostram as Figuras 46 e 47.

Figura 46 – Custo por Quilômetro Rodado.

km/l munck	custo diesel 07/10	custo/km
4,2	4,616	R$ 19,39

Fonte: Autor (2021).

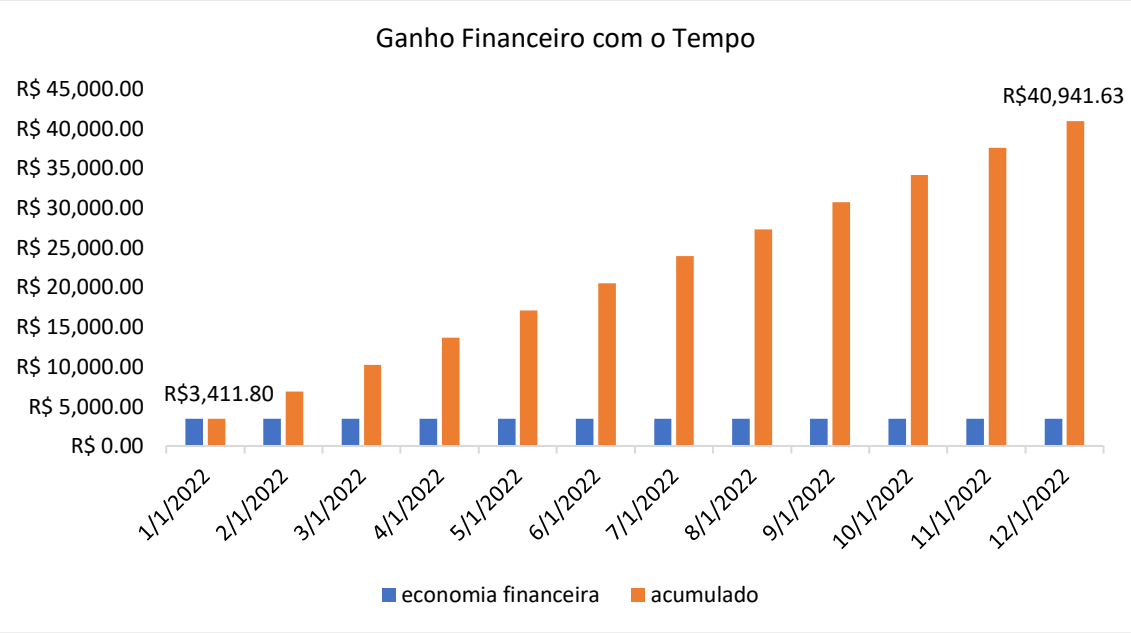
Figura 47 – Economia Financeira por Semana, Mês e Ano.

economia financeira obtida					
rotas/semana	5				
economia/semana	R$ 785,18		1 mês	4,34524	semanas
economia/mês	R$ 3.411,80		1 ano	52,1429	semanas
economia/ano	R$ 40.941,65				

Fonte: Autor (2021).

O Gráfico 1 mostra as projeções da economia de forma a simular a implementação por um ano, mês a mês iniciando em janeiro de 2022.

Gráfico 1 - Projeção da Economia Financeira.



Fonte: Autor (2021).

Além disso, a economia ambiental traz uma informação para o impacto da nova coleta na emissão de gases. Os valores utilizados para estimar as emissões foram provenientes do estudo de Manfrinato, Vidal e Brancalion (2021), em que cada litro de diesel corresponde a 2,96 quilogramas de dióxido de carbono, a partir do fator de transformação em CO2 e descontando o valor de etanol presente.

Dessa forma, foi realizado o cálculo da economia ambiental obtida, a partir da emissão de kg/CO2 evitada a partir da queima do combustível, como mostra a Figura 48.

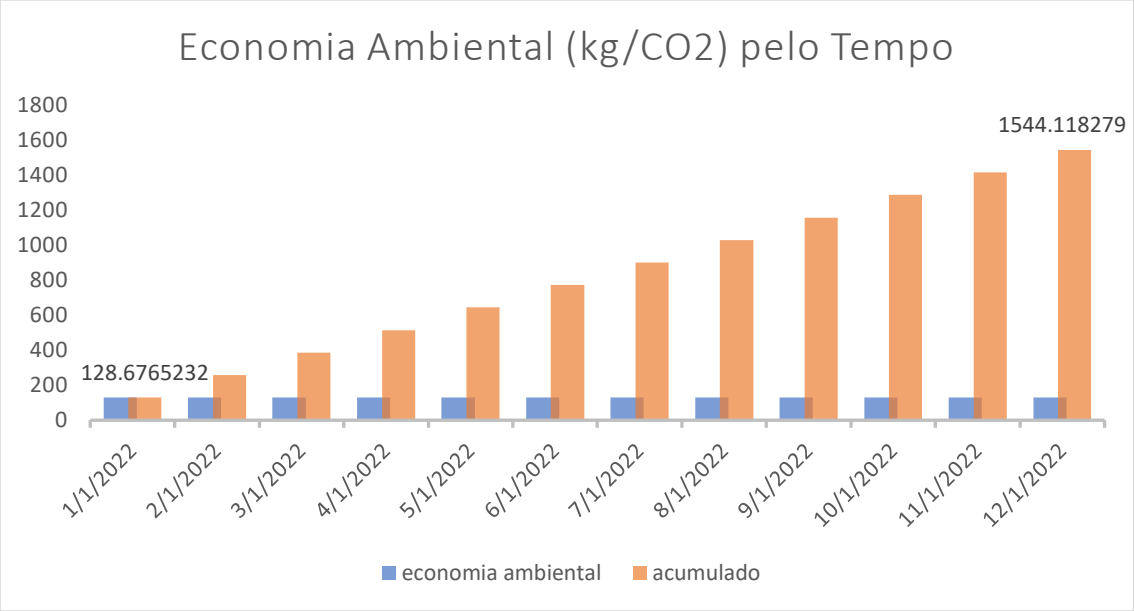
Figura 48 - Economia Ambiental por Semana, Mês e Ano.

	economia ambiental obtida		
	litros	kg/co2	
semana	9,642857143	29,61321429	
mês	41,90052857	128,6765232	
ano	502,8065357	1544,118871	

Fonte: Autor (2021).

Pode-se observar que a economia obtida pela roteirização é considerável quando projetada para períodos maiores, ainda considerando somente um caminhão no pátio da empresa. Foi realizada também a projeção da economia acumulada mês a mês seguindo o espaço de um ano, assim como a projeção financeira.

Gráfico 2 - Projeção da Economia Ambiental.



Fonte: Autor (2021).

A economia ambiental e financeiras acumuladas trazem uma visão de como o auxílio do uso de sensores pode auxiliar empresas que realizam a coleta seletiva, poupando mais de 40 mil reais e mais de 1,5 toneladas de dióxido de carbono despejados na atmosfera por ano. As conclusões finais estão no capítulo 6.

6 CONSIDERAÇÕES FINAIS, LIMITAÇÕES E FUTURAS LINHAS DE PESQUISA

Este capítulo tem como objetivo apresentar as conclusões do trabalho, além de suas limitações e trabalhos complementares que podem ser realizados.

6.1 CONSIDERAÇÕES FINAIS

O trabalho explorou a roteirização de uma empresa que realiza a coleta seletiva e tem interesse em tornar a coleta mais econômica, ecológica e inteligente a partir da utilização de sensores. Para isso, foram levantados os dados-chave do negócio, foi escolhido um *software* de otimização atualizado e adequado para o tipo de problema de logística. O *software* gerou rotas e foi possível comparar os custos de uma rota com ou sem o uso de sensores, permitindo a extrapolação dos resultados para a esfera financeira e ambiental.

6.2 LIMITAÇÕES

Como apresentado nos capítulos 4 e 5, o trabalho utilizou dados históricos, mas não trabalhou com dados em tempo real coletados por sensores permitindo uma análise preditiva. Ressalta-se o ponto que os valores trazidos de economia não podem ser generalizados e estão sujeitos a um caso específico, o que poderia ser mais fidedigno com um estudo global com mais rotas, empresas, número de pontos, materiais, o que pode aumentar ou diminuir os ganhos.

Foi utilizada uma amostra pequena de rotas, e para que o trabalho possua maior grau de fidelidade com a realidade, é indicado fazer um estudo com uma amostra maior que pode possibilitar análises como sazonalidade e previsão de ocupação dos contêineres a partir de modelos de aprendizado de máquina para séries temporais. Uma possibilidade é a utilização do método ARIMA e outras técnicas de previsão, como exibido por Dos Santos *et al.* (2019).

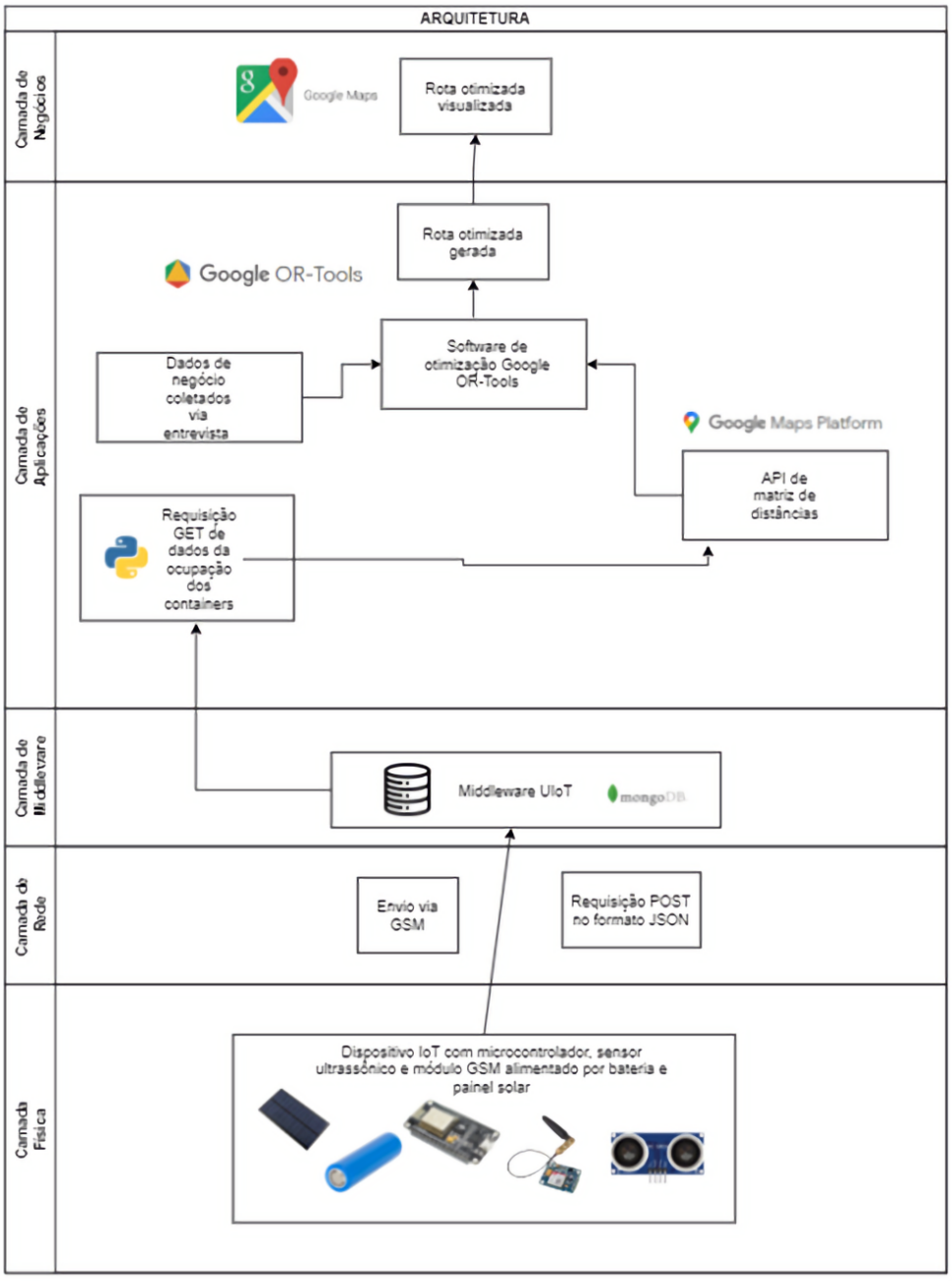
Para estimar o retorno para a empresa da instalação de um sistema IoT de *software* e hardware para coletar, monitorar e tratar os dados é necessário incluir gastos de manutenção, projeto, aquisição de equipamentos, hospedagem da aplicação, equipamentos e economias como manutenção dos caminhões, custo de mão de obra e outros fatores. Por isso, deve ser realizada uma análise de viabilidade econômica que leve

em conta todos os fatores necessários para fornecer indicadores financeiros que auxiliem na tomada de decisões.

6.3 TRABALHOS FUTUROS

A infraestrutura de *software* está preparada para receber os dados a partir do consumo do *middleware* UIoT, a diferença seria a inserção dos dados na planilha de acordo com os identificadores dos contêineres. A Figura 49 apresenta como seria a implementação ideal, dividida em camadas da arquitetura IoT:

Figura 49 - Arquitetura Sugerida de Trabalho Futuro.



Fonte: Autor (2021).

Nota-se que a arquitetura seria semelhante à utilizada em Da Luz *et al.* (2021, no prelo), mantendo-se as camadas física, de redes e de *middleware* e alterando a de aplicações e negócios. Os dados são coletados na camada física pelo dispositivo IoT e enviados via GSM para o *middleware*. Na camada de aplicações, o *script*, utilizando a linguagem de programação *Python*, faria o consumo dos dados do *middleware* para gerar as capacidades de cada contêiner e servir como *input* para a planilha utilizada pelo Google OR-Tools. Outros *inputs* para o *software* de otimização são os dados de negócio coletados como a capacidade do caminhão e os dados da API de matriz de distâncias a partir da latitude e longitude dos contêineres.

Finalmente, o Google OR-Tools geraria a rota otimizada de acordo com a localização, capacidade e restrições do modelo para os contêineres selecionados. Para melhorar a visualização, na camada de negócios seria gerada uma representação visual para o gestor de qual rota deve ser tomada.

Para que o projeto possa ser acompanhado e atualizado, é necessário hospedar a aplicação que realiza as rotas e o processamento e tratamento dos dados. Uma possibilidade de visualização é mostrar as rotas do dia a partir do *dashboard* já utilizado pela empresa para acompanhar a bateria e ocupação dos contêineres, exibido em Da Luz *et al.* (2021, no prelo). A partir desse *dashboard*, os gestores conseguem obter uma visão geral da situação de cada contêiner e quais rotas devem ser realizadas por data, além de um controle histórico automatizado.

Os dados coletados pelo dispositivo desenvolvido por Hellmers *et al.* (2020) podem ser integrados ao *middleware* UIoT, trazendo um volume maior de informações para complementar o trabalho e promovendo a multidisciplinaridade internacional no desenvolvimento de soluções tecnológicas.

Outra visualização que pode ser relevante para a empresa é uma interface com o celular para que o motorista possa visualizar em seu aplicativo de rotas a que deve ser feita no dia, além da integração com o *software* de frotas utilizado pela empresa. O trabalho possui espaço para novos desenvolvimentos e é escalável para aplicação em diversos contextos, como já é realizado em cidades inteligentes.

REFERÊNCIAS

AARONSON, Scott; IS, P. Is P versus NP formally independent?. **Bulletin of the EATCS**, v. 81, n. 109-136, p. 70, 2003.

AAZAM, Mohammad et al. Cloud of Things: Integrating Internet of Things and cloud computing and the issues involved. In: **Proceedings of 2014 11th International Bhurban Conference on Applied Sciences & Technology (IBCAST) Islamabad, Pakistan, 14th-18th January, 2014.** IEEE, 2014. p. 414-419.

ABRELPE - Associação Brasileira de Empresas de Limpeza Pública, 2020. **Panorama dos Resíduos Sólidos no Brasil**. Disponível em: https://abrelpe.org.br/panorama-2020/. Acesso em: 12 mai. 2021.

ADAMCZYK, Paul et al. Rest and web services: In theory and in practice. In: **REST: from research to practice.** Springer, New York, NY, 2011. p. 35-57.

ANAGNOSTOPOULOS, Theodoros et al. A stochastic multi-agent system for Internet of Things-enabled waste management in smart cities. **Waste Management & Research**, v. 36, n. 11, p. 1113-1121, 2018.

ANN, Celine Wan Shi; IQBAL, Norariefah Mohd. Open application programming interface (API): a financial revolution. **Bank Negara Malaysia Quarterly Bulletin**, v. 4, n. 1, p. 51-57, 2017.

ASGHARI, Parvaneh; RAHMANI, Amir Masoud; JAVADI, Hamid Haj Seyyed. Internet of Things applications: A systematic review. **Computer Networks**, v. 148, p. 241-261, 2019.

ASHTON, Kevin et al. That ‘internet of things’ thing. **RFID journal**, v. 22, n. 7, p. 97-114, 2009.

ATZORI, Luigi; IERA, Antonio; MORABITO, Giacomo. The internet of things: A survey. **Computer networks**, v. 54, n. 15, p. 2787-2805, 2010.

BISHOP, Chris M. Neural networks and their applications. **Review of scientific instruments**, v. 65, n. 6, p. 1803-1832, 1994.

BODIN, Lawrence. **The state of the art in the routing and scheduling of vehicles and crews**. Office of Policy Research, Urban Mass Transportation Administration, 1981.

BOSNJAK, Zita; GRLJEVIC, Olivera; BOSNJAK, Sasa. CRISP-DM as a framework for discovering knowledge in small and medium sized enterprises' data. In: **2009 5th International Symposium on Applied Computational Intelligence and Informatics**. IEEE, 2009. p. 509-514.

BOTTA, Alessio et al. Integration of cloud computing and internet of things: a survey. **Future generation computer systems**, v. 56, p. 684-700, 2016.

BRASIL, Lei N° 12.305 de 02 de agosto de 2010 - Política Nacional de Resíduos Sólidos (PNRS).

BRASIL, Lei nº 13.709, de 14 de agosto de 2018, dispõe sobre a proteção de dados pessoais e altera a Lei nº 12.965, de 23 de abril de 2014 (Marco Civil da Internet).

BUJEL, Kamil et al. Solving high volume capacitated vehicle routing problem with time windows using recursive-DBSCAN clustering algorithm. **arXiv preprint arXiv:1812.02300**, 2018.

BURKARD, Rainer E.; DEĬNEKO, Vladimir G. On the traveling salesman problem with a relaxed Monge matrix. **Information Processing Letters**, v. 67, n. 5, p. 231-237, 1998.

CHAPMAN, Pete et al. CRISP-DM 1.0: Step-by-step data mining guide. **SPSS inc**, v. 9, p. 13, 2000.

CHIANG, Mei-Ling; LI, Yun-Chen. LyraNET: A zero-copy TCP/IP protocol stack for embedded systems. **Real-Time Systems**, v. 34, n. 1, p. 5-18, 2006.

CLARKE, Geoff; WRIGHT, John W. Scheduling of vehicles from a central depot to a number of delivery points. **Operations research**, v. 12, n. 4, p. 568-581, 1964.

CORDEAU, Jean-Francois et al. A guide to vehicle routing heuristics. **Journal of the Operational Research society**, v. 53, n. 5, p. 512-522, 2002.

COSTA, Luciano; CONTARDO, Claudio; DESAULNIERS, Guy. Exact branch-price-and-cut algorithms for vehicle routing. **Transportation Science**, v. 53, n. 4, p. 946-985, 2019.

DA LUZ, G.P.C.P. et al. Sistema De Internet Das Coisas para Monitoração de Dados e Tomada de Decisões quanto à Coleta Seletiva de Lixo em Cidades Inteligentes. In: **Atas das Conferências IADIS Ibero-Americanas WWW/Internet 2021 e Computação Aplicada 2021**. 2021. No prelo.

DANTZIG, George B.; RAMSER, John H. The truck dispatching problem. **Management science**, v. 6, n. 1, p. 80-91, 1959.

DEKKER, Rommert; BLOEMHOF, Jacqueline; MALLIDIS, Ioannis. Operations Research for green logistics–An overview of aspects, issues, contributions and challenges. **European journal of operational research**, v. 219, n. 3, p. 671-679, 2012.

DÍAZ-DÍAZ, Raimundo; MUÑOZ, Luis; PÉREZ-GONZÁLEZ, Daniel. Business model analysis of public services operating in the smart city ecosystem: The case of SmartSantander. **Future Generation Computer Systems**, v. 76, p. 198-214, 2017.

DO PRADO, Daniel S. et al. Design of a Fog Controller to Provide an IoT Middleware with Hierarchical Interaction Capability. In: **International Conference on Information Technology & Systems**. Springer, Cham, 2021. p. 280-292.

DONG, Gaifang; GUO, William W.; TICKLE, Kevin. Solving the traveling salesman problem using cooperative genetic ant systems. **Expert systems with applications**, v. 39, n. 5, p. 5006-5011, 2012.

DOS SANTOS, J. K et al. Aplicação de Métodos de Previsão de Demanda e Gestão de

Estoque em um Restaurante Japonês em Brasília-DF. In: **ENEGEP – Encontro Nacional de Engenharia de Produção**, 2019, Santos. Anais. Santos: Abepro, 2019.

ELSHAER, Raafat; AWAD, Hadeer. A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants. **Computers & Industrial Engineering**, v. 140, p. 106242, 2020.

EWOUD, 2020. **Create a distance matrix in Python with the Google Maps API**. Disponível em: https://strategyanalytics.medium.com/create-a-distance-matrix-in-python-with-the-google-maps-api-737dd0fc8081. Acesso em: 19 out. 2021.

FARIAS, André Rodrigo et al. Identificação, mapeamento e quantificação das áreas urbanas do Brasil. **Embrapa Gestão Territorial-Comunicado Técnico (INFOTECA-E)**, 2017.

FERREIRA, Hiro Gabriel Cerqueira et al. Proposal of a secure, deployable and transparent middleware for internet of things. In: **2014 9th Iberian conference on information systems and technologies (CISTI)**. IEEE, 2014. p. 1-4.

FUKASAWA, Ricardo et al. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. **Mathematical programming**, v. 106, n. 3, p. 491-511, 2006.

GARZON CASTRO, Karina Andrea; MONDRAGÓN GARCÍA, Sebastián. A vehicle routing application for retail delivery with open source tools. 2020.

GENDREAU, Michel; LAPORTE, Gilbert; POTVIN, Jean-Yves. Metaheuristics for the capacitated VRP. In: **The vehicle routing problem**. Society for Industrial and Applied Mathematics, 2002. p. 129-154.

GOLDEN, Bruce L.; ASSAD, Arjang A. OR forum—perspectives on vehicle routing: exciting new developments. **Operations Research**, v. 34, n. 5, p. 803-810, 1986.

GOOGLE CLOUD PLATFORM, 2021. **Google Cloud Platform APIs Dashboard**. Disponível em: https://console.cloud.google.com/apis. Acesso em: 24 out. 2021.

GREENGARD, Samuel. The internet of things. MIT press, 2015.

GROSS, Jonathan L.; YELLEN, Jay. **Handbook of graph theory**. CRC press, 2003.

HAJGHASEM, Mahtab et al. Optimal routing in supply chain aimed at minimizing vehicle cost and supply. **Procedia Economics and Finance**, v. 36, p. 353-362, 2016.

HANNAN, M. A. et al. Capacitated vehicle-routing problem model for scheduled solid waste collection and route optimization using PSO algorithm. **Waste management**, v. 71, p. 31-41, 2018.

HASLE, Geir; KLOSTER, Oddvar. Industrial vehicle routing. In: **Geometric modelling, numerical simulation, and optimization**. Springer, Berlin, Heidelberg, 2007. p. 397-435.

HASSAN, Qusay F. A Tutorial Introduction to IoT Design and Prototyping with Examples. 2018.

HELLMERS, J. K.; LARSEN, K. H. K.; CHRISTIANSEN, M. H.; NIELSEN, R. N.. LoRa communication in waste collection: An international collaboration project. Aalborg Universitet project library, 2020. https://projekter.aau.dk/ projekter/da/studentthesis/.

HILLIER, F. S.; LIEBERMAN, G. J. Introduction to Operations Research, McGraw Hill. **Inc. New York**, p. 4-15, 1995.

IBGE - INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. Pesquisa Nacional por Amostra de Domicílios: Síntese de Indicadores. 2015. **IBGE**, Coordenação de Trabalho e Rendimento. - Rio de Janeiro: IBGE, 2016.108p.

JIA, Xiaolin et al. RFID technology and its applications in Internet of Things (IoT). In: **2012 2nd international conference on consumer electronics, communications and networks (CECNet)**. IEEE, 2012. p. 1282-1285.

KHAN, Rafiullah et al. Future internet: the internet of things architecture, possible applications and key challenges. In: **2012 10th international conference on frontiers of information technology**. IEEE, 2012. p. 257-260.

KOVAL, L.; VAŇUŠ, J.; BILÍK, P. Distance measuring by ultrasonic sensor. **IFAC-PapersOnLine**, v. 49, n. 25, p. 153-158, 2016.

KRISTANTO, Slamet et al. Dynamic polling algorithm for low energy garbage level measurement in smart trash bin. In: **Proceedings of the Second International Conference on IoT in Urban Space**. 2016. p. 92-94.

KUMAR, J. Sathish; PATEL, Dhiren R. A survey on internet of things: Security and privacy issues. **International Journal of Computer Applications**, v. 90, n. 11, 2014.

LAPORTE, Gilbert et al. Classical and modern heuristics for the vehicle routing problem. **International transactions in operational research**, v. 7, n. 4-5, p. 285-300, 2000.

LAPORTE, Gilbert; MARTELLO, Silvano. The selective travelling salesman problem. **Discrete applied mathematics**, v. 26, n. 2-3, p. 193-207, 1990.

LARSEN, Anna W.; MERRILD, Hanna; CHRISTENSEN, Thomas H. Recycling of glass: accounting of greenhouse gases and global warming contributions. **Waste Management & Research**, v. 27, n. 8, p. 754-762, 2009.

LE, Thai Quang; PISHVA, Davar. Optimization of convenience stores' distribution system with web scraping and Google API service. In: **2015 17th International Conference on Advanced Communication Technology (ICACT).** IEEE, 2015. p. 596-606.

LI, Bo Hu et al. Simulation on vehicle routing problems in logistics distribution. **COMPEL-The international journal for computation and mathematics in electrical and electronic engineering**, 2009.

LI, Shancang; DA XU, Li; ZHAO, Shanshan. The internet of things: a survey. **Information Systems Frontiers**, v. 17, n. 2, p. 243-259, 2015.

LI, Xiaoyan. **Capacitated Vehicle Routing Problem with Time Windows: A Case Study on Pickup of Dietary Products in Nonprofit Organization**. Arizona State University, 2015.

LYSGAARD, Jens; LETCHFORD, Adam N.; EGLESE, Richard W. A new branch-and-cut algorithm for the capacitated vehicle routing problem. **Mathematical Programming**, v. 100, n. 2, p. 423-445, 2004.

LUZ, GUSTAVO, 2021. IoT_Waste_Collection. **Zenodo**, out 2021. Disponível em: https://doi.org/10.5281/zenodo.5573172. Acesso em: 16 out. 2021.

MA, Wenqing; ZHANG, Jing. The survey and research on application of cloud computing. In: **2012 7th International Conference on Computer Science & Education (ICCSE)**. IEEE, 2012. p. 203-206.

MADEN, Will; EGLESE, Richard; BLACK, Dan. Vehicle routing and scheduling with time-varying data: A case study. **Journal of the Operational Research Society**, v. 61, n. 3, p. 515-522, 2010.

MANFRINATO, W, VIDAL, Edson, BRANCALION, Pedro, 2021. Como calcular as emissões de Gases de Efeito Estufa (GEE) produzidas no transporte de veículos?. **ESALQ / USP - Escola Superior de Agricultura Luiz de Queiroz**. Disponível em: http://esalqlastrop.com.br/capa.asp?pi=calculadora_emissoes. Acesso em: 17 out. 2021.

MARTÍNEZ-PLUMED, Fernando et al. CRISP-DM twenty years later: From data mining processes to data science trajectories. **IEEE Transactions on Knowledge and Data Engineering**, 2019.

MEISEL, Stephan; MATTFELD, Dirk. Synergies of operations research and data mining. **European Journal of Operational Research**, v. 206, n. 1, p. 1-10, 2010.

MENEZES, João Tribouillet Marcial de; COSTA, Pedro Henrique Lira da. Desenvolvimento de modelo hierárquico de middlewares com aplicação de Edge Computing para redes IoT. 2019.

MES, Martijn; SCHUTTEN, Marco; RIVERA, Arturo Pérez. Inventory routing for dynamic waste collection. **Waste management**, v. 34, n. 9, p. 1564-1576, 2014.

MILEVA, Yana Momchilova; DALLMEIER, Valentin; ZELLER, Andreas. Mining API popularity. In: **International Academic and Industrial Conference on Practice and Research Techniques**. Springer, Berlin, Heidelberg, 2010. p. 173-180.

MINIZINC, 2020. **Minizinc Challenge 2020 Results**. Disponível em: https://www.minizinc.org/challenge2020/results2020.html. Acesso em: 12 mai. 2021.

MIRJALILI, Seyedali. Genetic algorithm. In: **Evolutionary algorithms and neural networks**. Springer, Cham, 2019. p. 43-55.

MIYACHI, Christine. What is “Cloud”? It is time to update the NIST definition?. **IEEE Cloud computing**, v. 5, n. 03, p. 6-11, 2018.

MOHAMMED, Chnar Mustafa et al. Sufficient comparison among cloud computing

services: IaaS, PaaS, and SaaS: A review. **International Journal of Science and Business**, v. 5, n. 2, p. 17-30, 2021.

MONTEIRO, Matheus S. et al. Solid waste management and monitoringsystem for smart cities: development of alow-cost sustainable IoT architecture usingGPRS/GSM. In: **2021 Workshop on Communication Networks and Power Systems (WCNPS)**. IEEE, 2021. No prelo.

MURUGAANANDAM, S.; GANAPATHY, V.; BALAJI, R. Efficient IOT based smart bin for clean environment. In: **2018 International Conference on Communication and Signal Processing (ICCSP).** IEEE, 2018. p. 0715-0720.

NOVOA, Clara; STORER, Robert. An approximate dynamic programming approach for the vehicle routing problem with stochastic demands. **European journal of operational research**, v. 196, n. 2, p. 509-515, 2009.

OR-Tools, 2021. **CVRP Example**. Disponível em: https://developers.google.com/optimization/routing/cvrp. Acesso em: 19 out. 2021.

OTA, Matheus Jun et al. Algoritmo de Branch-Cut-and-Price para o Problema do Roteamento de Veículos Capacitados. **Revista dos Trabalhos de Iniciação Científica da UNICAMP**, n. 26, 2018.

PANDAS, 2021. **Pandas Dataframe**. Disponível em: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html. Acesso em: 17 out. 2021.

PATRÃO, Rafael L. et al. Technological Solution Development During the COVID-19 Pandemic: a Case Study in an IoT Lab. In: **2020 IEEE Global Conference on Artificial Intelligence and Internet of Things (GCAIoT)**. IEEE, 2020. p. 1-6.

PAYNTER, Sarah; JACKSON, C. M. Re-used Roman rubbish: a thousand years of recycling glass. **Post-Classical Archaeologies**, v. 6, p. 31-52, 2016.

PEARL, Judea. Heuristics: intelligent search strategies for computer problem solving. 1984.

PERRON, Laurent. Operations research and constraint programming at google. In: **International Conference on Principles and Practice of Constraint Programming. Springer**, Berlin, Heidelberg, 2011. p. 2-2.

PYTHON. **Itertools**. 2021. Disponível em: https://docs.python.org/3/library/itertools.html. Acesso em: 24 out. 2021.

RALPHS, T. K.; GÜZELSOY, M.; MAHAJAN, A. SYMPHONY 5.2. 3 user's manual. **Online: www. coin-or. org/download/binary/SYMPHONY**, 2010.

REINELT, Gerhard. **The traveling salesman: computational solutions for TSP applications**. Springer, 2003.

ROMANYCIA, Marc HJ; PELLETIER, Francis Jeffry. What is a heuristic?. **Computational Intelligence**, v. 1, n. 1, p. 47-58, 1985.

SAMANN, Fady EF. The design and implementation of smart trash bin. **Academic Journal of Nawroz University**, v. 6, n. 3, p. 141-148, 2017.

SAMPAIO, João Carlos V, 2015. **Passeios de Euler e as pontes de Königsberg**. Disponıvel em http://www. dm. ufscar. br/sampaio/PasseiosdeEuler. Pdf. Acesso em: 10 jul. 2021.

SAWANT, Anand Ashok; BACCHELLI, Alberto. A dataset for API usage. In: **2015 IEEE/ACM 12th Working Conference on Mining Software Repositories**. IEEE, 2015. p. 506-509.

SEHRAWAT, Deepti; GILL, Nasib Singh. Smart sensors: Analysis of different types of IoT sensors. In: **2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)**. IEEE, 2019. p. 523-528.

SHAHID, Noman; ANEJA, Sandhya. Internet of Things: Vision, application areas and research challenges. In: **2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)**. IEEE, 2017. p. 583-587.

SINGH, Dhananjay; TRIPATHI, Gaurav; JARA, Antonio J. A survey of Internet-of-Things: Future vision, architecture, challenges and services. In: **2014 IEEE world forum on Internet of Things (WF-IoT)**. IEEE, 2014. p. 287-292.

SLU. Serviço de Limpeza Urbana do Distrito Federal, 2020. **ampliação da coleta seletiva e impactos da pandemia nos serviços de limpeza urbanos**. Disponível em: https://www.slu.df.gov.br/wp-content/uploads/2021/03/RELATORIO-ANUAL-2020.pdf. Acesso em: 18 out. 2021.

SONI, Anshu; RANGA, Virender. API features individualizing of web services: REST and SOAP. **International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN**, p. 2278-3075, 2019.

STOPKA, Ondrej; JEŘÁBEK, Karel; STOPKOVÁ, Mária. Using the operations research methods to address distribution tasks at a city logistics scale. **Transportation Research Procedia**, v. 44, p. 348-355, 2020.

SURANA, Pratik. Benchmarking Optimization Algorithms for Capacitated Vehicle Routing Problems. 2019.

TALAVERA, Jesús Martín et al. Review of IoT applications in agro-industrial and environmental fields. **Computers and Electronics in Agriculture**, v. 142, p. 283-297, 2017.

TANIGUCHI, Eiichi; THOMPSON, Russell G. (Ed.). **City logistics: Mapping the future**. CRC Press, 2014.

TATROUS, Adell; SVENSSON, Rasmus. Design Abstraction of IoT REST APIs: Defining Design Patterns. 2020.

THONG, Stephen Teang Soo; HAN, Chua Tien; RAHMAN, Tharek Abdul. Intelligent fleet management system with concurrent GPS & GSM real-time positioning technology.

In: **2007 7th international conference on its telecommunications**. IEEE, 2007. p. 1-6.

TOTH, Paolo; VIGO, Daniele. Models, relaxations and exact approaches for the capacitated vehicle routing problem. **Discrete Applied Mathematics**, v. 123, n. 1-3, p. 487-512, 2002.

UNDP, 2015. **Sustainable Development Goals**. Disponível em: https://www.undp.org/sustainable-development-goals. Acesso em: 17 out. 2021.

WIGDERSON, Avi. P, NP and mathematics–a computational complexity perspective. In: **Proceedings of the ICM**. 2006. p. 665-712.

WINSTON, Wayne L. An introduction to model building. **Operations research applications and algorithms**, p. 2-5, 2004.

WIRTH, Rüdiger; HIPP, Jochen. CRISP-DM: Towards a standard process model for data mining. In: **Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining**. London, UK: Springer-Verlag, 2000.

WU, Miao et al. Research on the architecture of Internet of Things. In: **2010 3rd international conference on advanced computer theory and engineering (ICACTE)**. IEEE, 2010. p. V5-484-V5-487.

W3SCHOOLS, 2021. **Python Dictionaries**. Disponível em: https://www.w3schools.com/python/python_dictionaries.asp. Acesso em: 24 out. 2021.

YUSOF, N. Mohd et al. Smart waste bin with real-time monitoring system. **International Journal of Engineering & Technology**, v. 7, n. 2.29, p. 725-729, 2018.

ZHMUD, V. A. et al. Application of ultrasonic sensor for measuring distances in robotics. In: **Journal of Physics: Conference Series**. IOP Publishing, 2018. p. 032189.

APÊNDICE A

QUESTIONÁRIO COM A EMPRESA

1. Quantas vezes por semana ocorre a coleta?
2. Quanto um caminhão consegue carregar?
3. Todos os caminhões possuem a mesma capacidade de carga? Normalmente um caminhão visita quantos contêineres?
4. Quanto tem de vidro em kg em um contêiner?
5. Quantos caminhões há no pátio?
6. Quantos contêineres existem? quantos vão existir? quais as localizações precisas de cada um deles (latitude e longitude)?
7. Na hora de coletar, qual costuma ser a situação dos contêineres? Varia muito? Alguns muito cheios e outros muito vazios?
8. Qual o gasto médio mensal com combustível da frota (em litros ou em reais)? Quais os custos extras podem ser listados (manutenção dos caminhões por exemplo)?

APÊNDICE B

CÓDIGO DE ROTEIRIZAÇÃO

```
"""Capacited Vehicles Routing Problem (CVRP)."""

# version for 1 truck visiting 10+ places

from ortools.constraint_solver import routing_enums_pb2
from ortools.constraint_solver import pywrapcp
import pandas as pd
import googlemaps
from itertools import tee
import config
import time
from itertools import islice

#input: CSV file with id,latitude, longitude and capacities
# desired output: list with matrix distance for each point consumed by or
tools
start1 = time.time()

df = pd.read_excel('/home/gustavo/python/pg/rota_antiga.xlsx')

API_key = config.api_key #enter your google maps api key here
gmaps = googlemaps.Client(key=API_key)

#empty list - will be used to store calculated distances
time_list = []
distance_list = []
origin_id_list = []
destination_id_list = []

for (i1, row1) in df.iterrows():
  LatOrigin = row1['latitude']
  LongOrigin = row1['longitude']
  origin = (LatOrigin, LongOrigin)
  origin_id = row1['ID']
  for (i2, row2) in  df.iterrows():
    LatDestination = row2['latitude']
    LongDestination = row2['longitude']
    destination_id = row2['ID']
    destination = (LatDestination, LongDestination)
    result = gmaps.distance_matrix(origin, destination, mode='driving')
    result_distance =
result["rows"][0]["elements"][0]["distance"]["value"]
    result_time = result["rows"][0]["elements"][0]["duration"]["value"]
    time_list.append(result_time)
    distance_list.append(result_distance)
```

```
    origin_id_list.append(origin_id)
    destination_id_list.append(destination_id)

size=(len(df.coordinates))

# Input list initialization
Input = distance_list

# list of length in which we have to split
number_containers = len(df['ID'])
print("Number of Containers: ",number_containers)

length_to_split = number_containers*[number_containers]

# Using islice
Inputt = iter(Input)
Output = [list(islice(Inputt, elem))
          for elem in length_to_split]


stop1 = time.time()
start2 = time.time()
# Printing Output
print("API list", Input)
print("List of Lists", Output)
def create_data_model():
    """Stores the data for the problem."""
    data = {}

    #always 1 number bigger, because it has to return to base
    data['distance_matrix'] = Output

    # full container: 600 to 800kg
    capacities = df['PESAGENS']
    print(capacities)

    data['demands'] = df['PESAGENS'].values.tolist()
    print(data['demands'])

    # 1 vehicle
    # capacity: 9000 to 12000 kg
    data['vehicle_capacities'] = [10000]
    data['num_vehicles'] = 1

    data['depot'] = 0
    print("Entrada OR-Tools: ",data," ",type(data))
    return data


def print_solution(data, manager, routing, solution):
    """Prints solution on console."""
```

```
    print(f'Objective: {solution.ObjectiveValue()}')
    total_distance = 0
    total_load = 0
    for vehicle_id in range(data['num_vehicles']):
        index = routing.Start(vehicle_id)
        plan_output = 'Route for vehicle {}:\n'.format(vehicle_id)
        route_distance = 0
        route_load = 0
        while not routing.IsEnd(index):
            node_index = manager.IndexToNode(index)
            route_load += data['demands'][node_index]
            plan_output += ' {0} Load({1}) -> '.format(node_index,
route_load)
            previous_index = index
            index = solution.Value(routing.NextVar(index))
            route_distance += routing.GetArcCostForVehicle(
                previous_index, index, vehicle_id)
        plan_output += ' {0}
Load({1})\n'.format(manager.IndexToNode(index),
                                                route_load)
        plan_output += 'Distance of the route:
{}m\n'.format(route_distance)
        plan_output += 'Load of the route: {}\n'.format(route_load)
        print(plan_output)
        total_distance += route_distance
        total_load += route_load

    print('Total distance of all routes: {}m'.format(total_distance))
    print('Total load of all routes: {}'.format(total_load))


def main():
    """Solve the CVRP problem."""
    # Instantiate the data problem.
    data = create_data_model()

    # Create the routing index manager.
    manager = pywrapcp.RoutingIndexManager(len(data['distance_matrix']),
                                           data['num_vehicles'],
data['depot'])

    # Create Routing Model.
    routing = pywrapcp.RoutingModel(manager)


    # Create and register a transit callback.
    def distance_callback(from_index, to_index):
        """Returns the distance between the two nodes."""
        # Convert from routing variable Index to distance matrix
NodeIndex.
        from_node = manager.IndexToNode(from_index)
```

```
        to_node = manager.IndexToNode(to_index)
        return data['distance_matrix'][from_node][to_node]

    transit_callback_index =
routing.RegisterTransitCallback(distance_callback)

    # Define cost of each arc.
    routing.SetArcCostEvaluatorOfAllVehicles(transit_callback_index)


    # Add Capacity constraint.
    def demand_callback(from_index):
        """Returns the demand of the node."""
        # Convert from routing variable Index to demands NodeIndex.
        from_node = manager.IndexToNode(from_index)
        return data['demands'][from_node]

    demand_callback_index = routing.RegisterUnaryTransitCallback(
        demand_callback)
    routing.AddDimensionWithVehicleCapacity(
        demand_callback_index,
        0,  # null capacity slack
        data['vehicle_capacities'],  # vehicle maximum capacities
        True,  # start cumul to zero
        'Capacity')

    # Setting first solution heuristic.

    search_parameters = pywrapcp.DefaultRoutingSearchParameters()
    search_parameters.first_solution_strategy = (
        routing_enums_pb2.FirstSolutionStrategy.AUTOMATIC)
    search_parameters.local_search_metaheuristic = (
        routing_enums_pb2.LocalSearchMetaheuristic.GUIDED_LOCAL_SEARCH)
    search_parameters.time_limit.FromSeconds(1)

    # Solve the problem.
    solution = routing.SolveWithParameters(search_parameters)

    # Print solution on console.
    if solution:
        print_solution(data, manager, routing, solution)
    else:
        print("no solution found")


if __name__ == '__main__':
    main()
    stop2 = time.time()
    print("Time to run API:", stop1-start1)
    print("Time to run OR-Tools:", stop2-start2)
```