Universidade de Brasília

Faculdade de Economia, Administração, Contabilidade e Gestão de Políticas Públicas - FACE

Departamento de Administração

Victor Costa Caldas

Automação em melhoria de processos: desenvolvimento tecnológico voluntário em logística urbana para uma empresa brasileira de medicina

Brasília - DF

2021

Victor Costa Caldas

Automação em melhoria de processos: desenvolvimento tecnológico voluntário em logística urbana para uma empresa brasileira de medicina

Monografia apresentada ao Departamento de Administração da Universidade de Brasília como requisito parcial à obtenção do título de Bacharel em Administração.

Professor Orientador: Msc. Cayan Atreio Portela Bárcena Saavedra

Brasília - DF

2021

Victor Costa Caldas

Automação em melhoria de processos: desenvolvimento tecnológico voluntário em logística urbana para uma empresa brasileira de medicina

A Comissão Examinadora, abaixo identificada, aprova o Trabalho de Conclusão do Curso de Administração da Universidade de Brasília do aluno

Victor Costa Caldas

Msc. Cayan Atreio Portela Bárcena Saavedra
Professor-Orientador

Dr. Peng Yaohao
Professor-Examinador

Msc. Eloísa Gonçalves da Silva Torlig
Professor-Examinador

Brasília, 18 de maio de 2021
semestre letivo 2/2020

Agradecimentos

Agradeço imensamente a minha mãe e meu pai por todo o esforço em me criar feliz e saudável; aos meus avós, meus irmãos, meus primos e meus tios pelo imutável amor e apoio; aos meus amigos verdadeiros pelos ensinamentos e companhia de sempre; ao meu orientador pela confiança e atenção; às instituições EdN e UnB pela minha formação e capacitação. Agradeço também a todos os colegas e pessoas que me acompanharam nessa caminhada, por servirem de exemplo para me tornar uma pessoa melhor.

Dedicatória

Dedico este trabalho a toda a minha família. Sobretudo ao meu primo Ricardinho pela permanente participação na minha paixão por computação, companheirismo e constante auxílio nas ideias e desenvolvimentos, e à minha madrinha, tia Sandra, pelo grande exemplo em administração de empresas, excelentes oportunidades ao longo da minha jornada e eterno carinho.

Epígrafe

Administração de empresas é como um software que roda em um ambiente levemente aleatório. Você os cria baseando-se em expectativas comprováveis, e os modifica conforme oportunidades de melhoria são identificadas.

Managing a business is like setting a software to run in a slightly random environment. You create them based on provable expectations, and modify them as enhancement opportunities are identified.

Victor C. Caldas

Motivação para este trabalho

A automação é uma das vertentes tecnológicas com maior impacto em prol da produtividade da humanidade e existem cada vez mais meios de efetuá-la. É oportuno finalizar esta etapa da minha trajetória acadêmica e progredir na trajetória profissional e pessoal auxiliando uma empresa nacional a se desenvolver tecnologicamente por meio da automação, tornando-a mais atualizada e competitiva nacionalmente e globalmente.

Aplicar como tema do trabalho de conclusão de curso uma melhoria de processos corporativos através da programação, ferramenta que está em processo crescente de descoberta pelo mercado e tem uma infinidade de aplicações úteis para todas as áreas, é uma grande oportunidade. Pois a partir da prática, foi possível ir além de ajudar uma organização, mas aprender mais sobre problemas de otimização, bancos de dados e estruturação de algoritmos. Trazer o tema à pauta acadêmica e institucional de forma acessível e com uma linguagem compreensível tem o potencial de desencadear em outras pessoas a busca por resoluções de demais problemas interessantes e relevantes. Almejo que a divulgação deste projeto e os métodos exercidos agreguem na visão do leitor sobre possibilidades de aplicações, utilidade e a importância da programação no contexto mundial atual.

Resumo

O surgimento dos computadores pessoais e celulares inteligentes gerou uma revolução na utilização da internet, na acessibilidade individual à informação e nos processos corporativos. Hoje observamos uma nova revolução: os meios tecnológicos voltados à automação de processos e robotização de operações cognitivas e manuais vêm se ampliando de forma acelerada, e já é possível perceber alguns impactos que ocorrerão nos próximos anos. Este trabalho consiste na apresentação da formulação e avaliação de uma dessas tecnologias, construída voluntariamente por meio da programação, como produto deste projeto, para o departamento de logística urbana de uma empresa de medicina nacional. O processo automatizado envolve a roteirização de veículos urbanos, resolvendo o problema computacional de otimização combinatória chamado de Problema de Roteamento de Veículos (PRV). A automação desse processo tem efeito principalmente sobre os objetivos corporativos financeiros, por meio da redução de custos e do aumento da capacidade produtiva, mas também atinge positivamente outras frentes como a gestão por desempenho de RH, a qualidade do atendimento ao cliente, o controle sobre a operação e a redução de emissão por veículo ao meio ambiente. Os resultados, observados por meio de simulações, se provaram amplamente satisfatórios: o *software* possibilita uma redução de custos de aproximadamente R$78 mil mensais, com um aumento do faturamento de aproximadamente R$1,6 milhões mensais através da expansão da operação. A avaliação de eficiência do aplicativo por meio da pesquisa operacional utilizou mais de 100 blocos distintos de localidades aleatórias, cada um contendo de 28 a 264 endereços. Os ganhos foram verificados através das diferenças em cada bloco entre os desempenhos do formato manual da operação (grupo de controle) e do formato automático executado pelo algoritmo. A execução mecânica encontrou soluções melhores que o grupo de controle em todas as simulações.

Palavras-chave: automação, melhoria de processos, remodelagem, logística, programação, otimização combinatória, roteirização de veículos, processo corporativo.

Abstract

The emergence of personal computers and smartphones created a revolution on internet usage, individual accessibility to information and corporate processes. Today we observe a new revolution: the technological means aimed at process automation and robotization of cognitive and manual operations have been expanding at an accelerated rate, and it's already possible to perceive some impacts that will occur in the upcoming years. This work consists of presenting the formulation and evaluation of one of these technologies, built voluntarily through programming, as a product of this project, for the department of urban logistics in a Brazilian medicine company. The automated process involves the routing of urban vehicles, through solving the computational problem of combinatorial optimization called the Vehicle Routing Problem (VRP). The automation of this process has an effect mainly on the corporation's financial objectives, by reducing costs and increasing production capacity, but it also positively affects other fronts such as HR performance management, quality of customer service, control over operation, and reduction of environmental emission per vehicle. The results, observed through simulations, proved to be highly satisfactory: the software enables a cost reduction of approximately R$ 78 thousand per month, with an increase in revenue of approximately R$ 1.6 million per month through the expansion of the operation. The application's efficiency evaluation through operational research used more than 100 distinct blocks of random locations, each containing 28 to 264 addresses. The gains were verified through the differences in each block between the performances of it’s manual format of the operation (control group) and the automatic format performed by the algorithm. The mechanical execution found better solutions than the control group in all simulations.

Keywords: automation, process improvement, remodeling, logistics, programming, combinatorial optimization, vehicle routing, corporate processes.

Resumen

El levante de las computadoras personales y los teléfonos inteligentes ha generado una revolución en el uso de Internet, en la accesibilidad individual a la información y en los procesos corporativos. Hoy observamos una nueva revolución: los medios tecnológicos orientados a la automatización de procesos y la robotización de operaciones cognitivas y manuales se están expandiendo a un ritmo acelerado, y ya es posible percibir algunos impactos que se producirán en los próximos años. Este trabajo consiste en presentar la formulación y evaluación de una de estas tecnologías, construida voluntariamente mediante programación, como producto de este proyecto, al departamento de logística urbana de una empresa brasileña de medicina. El proceso automatizado implica el enrutamiento de vehículos urbanos, resolviendo el problema computacional de optimización combinatoria llamado Problema de Enrutamiento de Vehículos. La automatización de este proceso tiene un efecto principalmente sobre los objetivos financieros corporativos, al reducir costos y aumentar la capacidad productiva, pero también afecta positivamente otros frentes como la gestión del desempeño de RRHH, la calidad del servicio al cliente, el control de la operación y la reducción de emisiones por vehículo al medio ambiente. Los resultados, observados mediante simulaciones, se demostraron muy satisfactorios: el software permite una reducción de costos de aproximadamente R$ 78 mil mensuales, con un aumento de los ingresos de aproximadamente R$ 1,6 millones mensuales mediante la ampliación de la operación. La evaluación de la eficiencia de la aplicación por medio de la investigación operativa utilizó más de 100 bloques distintos de ubicaciones aleatorias, cada uno con entre 28 y 264 direcciones. Las ganancias se verificaron a través de las diferencias en cada bloque entre los desempeños del formato manual de la operación (grupo de control) y el formato automático realizado por el algoritmo. La ejecución mecánica encontró mejores soluciones que el grupo de control en todas las simulaciones.

Palabras clave: automatización, mejoría de procesos, perfeccionamiento, logística, programación, optimización combinatoria, enrutamiento de vehículos, proceso corporativo.

Lista de ilustrações

Lista de gráficos

Lista de tabelas

Lista de abreviaturas e siglas

PCV	Problema do Caixeiro Viajante
PRV	Problema de Roteamento de Veículos
PRVJT	Problema de Roteamento de Veículos com Janelas de Tempo
PRVC	Problema de Roteamento de Veículos Capacitados
PRVCJT	Problema de Roteamento de Veículos Capacitados com Janelas de Tempo
OSM	*Open Street Map*
OSRM	*Open Source Routing Machine*
ROI	*Return over investment* (Retorno sobre investimento)

Sumário

1. Introdução

Um dos pontos mais relevantes do desenvolvimento tecnológico moderno é o anseio dos países e empresas por progredirem, criarem novas ideias para o mundo e conquistarem parcelas maiores do mercado. O termo comum que define esta constante busca por superação e melhoria de instituições é chamado de competitividade. Enquanto alguns países buscam investir em educação e desenvolvimento, e estabelecer políticas públicas para suas empresas a fim de se posicionarem melhor na competitividade global, outros atribuem outros focos aos seus investimentos, fato evidenciado pelos índices históricos de dispêndios globais em Pesquisa e Desenvolvimento (Unesco – "*Despesas internas brutas em Pesquisa e Desenvolvimento como porcentagem do PIB*", 1996-2019), melhor abordados à frente. Países com níveis maiores de corrupção, por exemplo, gastam menos na área da educação em comparação a países mais honestos da mesma faixa econômica (Mauro et al., 2019).

A competitividade organizada originou-se nos primórdios da civilização, quando surgiram os primeiros esforços coordenados em busca de entregar bens e serviços úteis à comunidade. A concorrência empresarial, por sua vez, veio à tona no século 18 com a Revolução Industrial, época em que surgiram as primeiras empresas formais (Project Management Knowledge Base, 2017). Desde então, a prática da competição entre empresas vem se fortalecendo conforme o crescimento da população e da tecnologia, empurrada por cada vez mais pessoas e entidades capazes e dispostas a produzir e entregar algo que acreditam ser desejado, benéfico, ou necessário para a sociedade. Por outro lado, investimentos em tecnologia também são utilizados como o principal meio de isolamento de uma empresa como líder absoluta de um segmento de mercado (Forbes, 2020). Com tudo isso, algoritmos de automação e aprendizagem de máquina vêm se tornando cada vez mais comuns, principalmente devido à competição entre empresas em aumentar a produção e reduzir custos (Albuquerque et al., 2019) através da melhoria de processos. Portanto, é importante que os administradores se conscientizem dessas novas tecnologias e como suas empresas podem se beneficiar

da implementação de tais sistemas (Albuquerque et al., 2019) para que possam estabelecer operações cada vez mais eficientes em suas respectivas áreas de atuação.

Hoje em dia, como nunca antes, o desenvolvimento tecnológico organizacional é essencial para a manutenção do desempenho de mercado de empresas de qualquer ramo. Dados da *Global Innovation 1000*, índice das empresas que mais contribuem mundialmente em pesquisa e desenvolvimento, compilados pela *strategy&*, demonstram os investimentos em inovação em 2018 das grandes organizações do ramo de Farmácia e Biotecnologia: com mais de 120 anos de história, a suiça Roche e estadunidense Merck & Co. investiram respectivamente 18,9% do faturamento anual, equivalente a 10,8 bilhões de dólares, e 25,4%, equivalente e U$10,2 bilhões. A também estadunidense Celgene Corporation investiu 6 bilhões, equivalente a 45,5% do faturamento do ano. Esta, com “apenas” 35 anos de vida e bem mais nova que suas competidoras, tem uma posição especial na competitividade de mercado e precisa de um esforço maior para conquistar e firmar sua *market share*, por estar consolidada a bem menos tempo. Logo, nota-se que nas maiores e mais influentes empresas do mundo, figura entre os maiores gastos (em formato de investimentos) a atualização tecnológica (i.e. inovação). Isto porque o mundo corporativo atual, imerso no exponencial cenário científico-tecnológico em que vivemos, debruça-se fortemente em soluções, ferramentas, e inovações, numa constante busca por entregar o melhor produto ou serviço à maior quantidade de pessoas possível.

Especificamente no setor de logística, os investimentos dos EUA em 2018 foram de 8% do PIB do país, aponta o Departamento de Comércio dos Estados Unidos, o equivalente a 1,6 trilhões de dólares. O blog estadunidense *edesk* por sua vez aponta que em 2020, dentre os gastos com logística no país, 50,3% foram referentes às atividades de transporte. Em 2019, informa o blog, a média de custos com logística nas empresas foi equivalente a 11% dos ganhos. Portanto entende-se que, em média, as empresas estadunidenses gastam com transporte em torno de 5,5% de suas receitas totais.

O presente trabalho descreve o desenvolvimento tecnológico efetuado voluntariamente para o setor de logística de transportes de uma empresa nacional de medicina, na filial do Distrito Federal, e agrega os métodos utilizados à bibliografia

nacional, a fim de expandir o acesso ao conhecimento e o desenvolvimento do país. Para executar o projeto, foi utilizado como meio principal a programação de computadores (i.e. linguagem de código) para desenvolver e implementar um algoritmo sequencial adequado às necessidades e especificidades da empresa, a fim de reestruturar o processo já existente para um formato automatizado e mais controlado, acurado e eficiente.

As abordagens focadas para o desenvolvimento da solução descrita tiveram como intenções principais i) a melhoria da capacidade de monitoramento sobre a equipe de serviços móveis, ii) redução do esforço da equipe de retaguarda em tentar formular a logística mais adequada, iii) aumento do controle sobre tempos e distâncias dos percursos, a fim de adequar a oferta a toda a demanda disponível, iv) redução de custos via economia de gasolina, desvalorização de automóveis e recursos humanos, v) aumento da capacidade produtiva da equipe, podendo efetuar mais atendimentos em menos tempo, vi) redução de atrasos pela equipe de transportes através de restrições algorítmicas que previnem que limites de tempo previstos sejam extrapolados, vii) melhoria da satisfação e segurança do funcionário de transporte, não precisando de pressa para percorrer suas rotas, e viii) aumento da qualidade do serviço entregue ao cliente devido ao aumento da disponibilidade e da confiança. Para isso, estuda-se:

a) o aperfeiçoamento da divisão de grupos de localidades, referentes aos clientes que devem ser visitados pela equipe de serviço móvel da empresa;

b) aperfeiçoamento da ordem em que cada membro da equipe deve cobrir os endereços do grupo que lhe foi atribuído;

c) clareza e precisão na exposição dos dados envolvidos no processo, podendo a equipe de retaguarda tomar decisões com mais assertividade.

O alcance dos requisitos acima expostos se faz possível devido ao comportamento da linguagem de máquina, isto é, as diretrizes que um computador recebe em seu processador lógico para executar determinadas instruções. As informações são passadas pelo usuário a partir de linguagens de programação, criadas para permitir que o ser humano se comunique com a máquina através da interpretação

de textos legíveis e tradução para os números binários (zeros e uns). Diferentemente dos recursos humanos, os recursos tecnológicos são montados de forma que, uma vez organizado o ambiente, acertado o processo a ser seguido e estabelecidas todas as restrições e parâmetros que o sistema necessita, a taxa de erro praticamente se nulifica e a produção se torna estável, acessível e ininterrupta. O custo cai, pois passa a depender apenas da máquina, energia e às vezes internet, e o produto fica disponível indefinidamente. O software desenvolvido e exposto neste trabalho foi construído por meio da linguagem interpretada Python, resolvendo o problema computacional chamado de Problema de Roteamento de Veículos através de uma heurística (uma solução ótima ou provavelmente boa para o problema computacional, resolvida em um tempo de execução aceitável) para a otimização dos trajetos, utilizando-se, dentre outros, uma variação do método de particionamento k-médias (*k-means clustering*) e a ferramenta de código aberto *OSRM*.

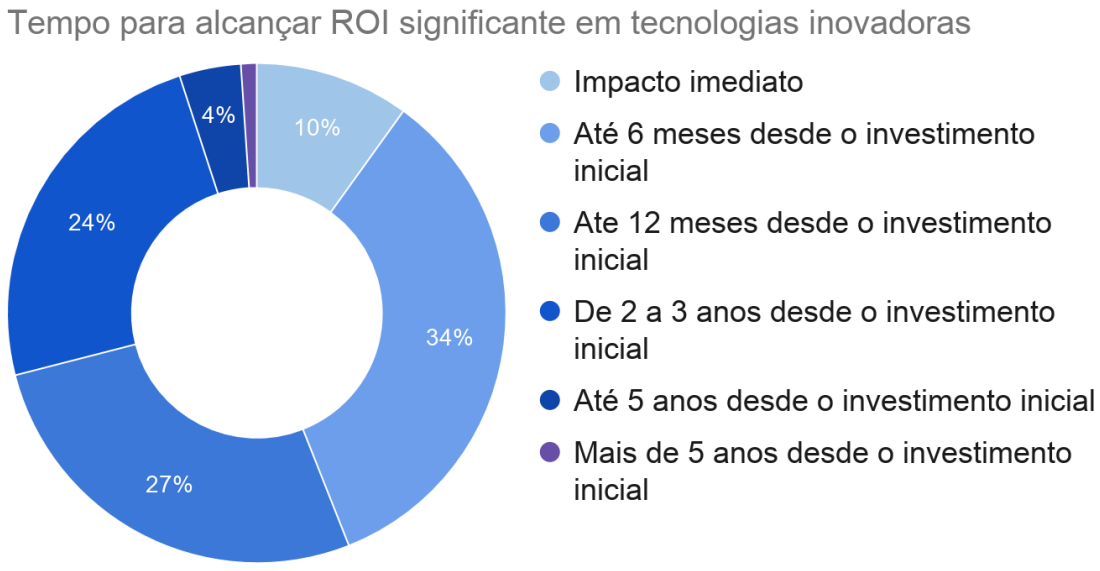
Dentre as grandes diferenças entre a execução de tarefas por computadores versus por seres humanos está a capacidade do aparelho em reproduzir rigorosamente as ordens que lhe são passadas. Além de ter uma memória mais eficaz que a nossa, a máquina consegue executar cálculos rápidos e livres de erro para atender às solicitações recebidas, desde que estas solicitações também sejam livres de erros. Por conta disso, a resolução de tarefas não-aleatórias costumam ser mais eficientes e econômicas a longo prazo quando executadas por computadores. Pessoas são mais resilientes e portanto se adequam a aleatoriedades que possam surgir durante um processo organizacional, e são mais inteligentes, logo requerem menos instruções, mas são mais imprecisas, lentas e indisponíveis.

A adequação do algoritmo à precisão nas instruções transmitidas à máquina constitui a base deste projeto, assim como corresponde à maior parte do tempo despendido em sua construção. No caso do problema aqui proposto, avaliações foram feitas para verificar as diferenças de resultados em divisões de rotas construídas entre 1) uma equipe de pessoas que têm prática e conhecem muito bem a cidade de Brasília e os endereços contidos nela, e 2) uma máquina rodando um algoritmo que consegue encontrar precisamente um ponto no mapa através de um dado endereço e consegue calcular rapidamente a exata distância de um ponto geográfico até outro. As avaliações

demonstram que existe um ganho muito significante ao utilizar a máquina para efetuar a tarefa.

Em termos de Retorno Sobre o Investimento, este projeto em inovação tecnológica é de caráter imediato: as contas da empresa sentem o impacto positivo a partir do momento em que o sistema é instaurado e utilizado, fato que posiciona o presente software na categoria dos 10% melhores investimentos em tecnologia inovativa para empresas, ao tratarmos da rapidez de retorno. A comprovação foi feita em pesquisa conduzida pela empresa KPMG em 2018 e 2019, intitulada “KPMG Pesquisa de Inovação na Indústria de Tecnologia 2019” (traduzido do inglês). Este braço da pesquisa foi denominado “Tempo para atingir retorno significante sobre o investimento em tecnologias inovadoras”, e os resultados mostram que menos de 1% das aplicações demoram mais de 5 anos para obter o retorno. Dos investimentos institucionais em tecnologias, 24% demoram entre 2 e 3 anos a partir da alocação inicial para retornar o valor alocado, 27% retornam o valor em até 12 meses, 34% retornam em até 6 meses e 10% retornam imediatamente.

Gráfico 1 – Tempo para alcançar ROI significante em tecnologias inovadoras.



Fonte: Reprodução traduzida própria de *KPMG Technology Industry Innovation Survey* 2019.

Neste caso específico, como o software foi construído em qualidade voluntária, faz-se evidente que o retorno seja imediato, tão logo o software seja implementado, já que não houve investimentos por parte da empresa. Ainda que fosse cobrado, o valor seria menor que o ganho mensal estimado, mantendo-se na casa dos primeiros 10% em rapidez de retorno.

2. Contextualização

O mundo enfrenta uma crescente no que trata de desenvolvimento tecnológico baseado em computadores. O século 21 trouxe consigo um forte aumento na produção de máquinas pessoais, diversas inovações em acesso à internet, e uma geração de dados diária sem precedentes. Desde o início do século até dezembro de 2020, o acesso à internet cresceu de 5,8% para 64,2% da população mundial, de acordo com dados do site *internetworldstats* que acompanha tendências mundiais da internet. A quantidade de dados circulantes mundialmente duplica a cada 2-3 anos, relata o site estatístico *statista*. A era dos computadores estabeleceu um marco histórico quando falamos de acessibilidade individual à informação e a ferramentas inteligentes, devido à expansão do conteúdo disponível e ao constante crescimento da base de usuários conectados à rede. De forma inevitável, sobe continuamente o número de pessoas aptas a desenvolverem alguma solução que contemple um problema mundano recorrente, e traz em si o aumento das ferramentas de código aberto (do inglês: *open source*), que disponibilizam o código-fonte gratuitamente a usuários interessados. O aumento da acessibilidade e a tendência de linguagens de programação serem cada vez mais compreensíveis e utilizadas vêm notadamente fazendo e permitindo com que a indústria seja revolucionada cada vez mais pela automação de atividades.

Conforme relatório construído em 2017 pela empresa McKinsey, baseado em estimativas do órgão *World Bank*, 30% das atividades de 6 a cada 10 empregos, ou pelo menos 18% de todas as atividades profissionais existentes mundialmente, podem ser automatizadas. O estudo mostra que a tendência até 2030 é que 375 milhões de trabalhadores ao redor do mundo mudem de categoria ocupacional, já que até lá muitas das áreas de atividades hoje comuns serão substituídas. No entanto, não é preciso temer a perda de empregos: as tendências históricas desde o início da industrialização sugerem que o mercado se ajusta às mudanças e novas oportunidades são geradas. Alexopoulos e Cohen (2016 apud Albuquerque et al., 2019) descobriram que choques tecnológicos historicamente aumentaram oportunidades de trabalho e empregabilidade em geral. Além disso, a rapidez desse câmbio em cada país depende de alguns fatores sobre sua capacidade em executar a transição, que podem ser

resumidos na posição que ele ocupa em relação aos outros, avaliada em PIB per capita. A quantidade prevista de empregos deslocados até 2030 no Brasil, por exemplo, é de 15,7 milhões, o que representa 7,3% da população, enquanto a previsão para o Japão, de 15,5 milhões, representa 12,5% da população. Ainda assim, o Brasil hoje tem suas atividades bem menos robotizadas que em países como o Japão, e portanto a proporção de atividades automatizáveis em âmbito nacional é maior: um estudo que avaliou dados de 1986 a 2017 indicou que, em 2017, 55% de todos os trabalhadores formalmente empregados nacionalmente estavam em empregos com riscos elevados ou muito elevados de serem substituídos por automação (Albuquerque et al., 2019). Em desenvolvimento acelerado, estima-se que a China terá, até 2049, 278 milhões de empregos substituídos por inteligência artificial, o equivalente a 35,8% dos empregos chineses atuais (Zhou, 2019).

Os investimentos em tecnologia feitos pelas empresas inseridas no S&P 500, um índice voltado a investimentos que compila desempenhos de ações das quinhentas maiores empresas das bolsas de valores NASDAQ e NYSE dos Estados Unidos, mostram que a alocação financeira em pesquisa e desenvolvimento (P&D) costuma compensar. Segundo dados agrupados e examinados por analista da empresa Sather Research e expostos em seu site, dentre todas as empresas do índice, a média de investimentos em P&D em 2020 foi de 4% do faturamento. O percentual tende a aumentar anualmente: nos últimos 10 anos (2011-2020), a média foi de 3,97%, e nos 10 anos anteriores a esses (2001-2010), 3,45% do faturamento. Alguns setores têm médias bem diferentes do restante: empresas imobiliárias tiveram um investimento médio de apenas 0,06% em pesquisa e desenvolvimento ao longo dos últimos 20 anos, enquanto o setor de saúde investiu 9,7% e empresas de tecnologia 11,19%. Nos nichos mais específicos das empresas do índice, dentre as que alocaram maior percentual em P&D em 2020, empresas de Biotecnologia lideram com aplicação média de 30,3% do faturamento. Empresas de Tecnologia em Saúde alocaram 13,6%, as farmacêuticas 15,2%, empresas de distribuição de equipamentos e materiais médicos 7,4%, e empresas de Software 19%.

A empresa de pesquisa FactSet obteve dados cujos posteriormente foram informados pela bolsa de valores NASDAQ, que indicam que dentre as empresas dos

EUA, os investimentos de 2018 em propriedade intelectual representavam à época um dos maiores valores da história no quesito. O setor privado executava no ano um aumento de 11,1% relativo ao ano anterior nos investimentos em propriedade intelectual, alocando até agosto de 2018 um montante total de 893 bilhões de dólares (5,14 trilhões de reais nos valores atuais), de acordo com a Secretaria de Análise Econômica (SAE) dos EUA. Inclusive, observa a SAE, que isso contribuiu por volta de 0,5% dentre os 3,1% de crescimento econômico do ano, ou seja, 16% do crescimento econômico do país no ano foi devido aos aportes em propriedade intelectual. Foi a maior contribuição dos investimentos em P&D do século para o país, e a tendência é aumentar. Este tipo de investimento dentre as grandes empresas de tecnologia, as chamadas *Big techs*, tem médias ainda maiores. Apenas nos 6 últimos meses de 2018, a Amazon investiu 14,01 bilhões de dólares em P&D, a Alphabet (holding do Google) investiu U$10,15 bilhões, e a Microsoft U$7,75 bilhões. Figuram também entre as que mais gastaram em P&D as empresas de medicina e saúde Merck, aplicando U$5,35 bilhões, Johnson & Johnson (U$4,79 bilhões), Bristol-Myers-Squibb (U$3,65 bilhões), Pfizer (U$3,53 bilhões) e Celgene (U$3,45 bilhões). Ainda de acordo com a bolsa de valores NASDAQ, cada uma dessas empresas está tentando maximizar suas parcelas de mercado por meio de melhorias internas de processos, serviços e produtos, investindo em práticas e tecnologias emergentes. Segundo os cientistas da bolsa estadunidense, são as pesquisas que fazemos hoje que garantirão uma entrega de qualidade de produtos e serviços no futuro. Altos investimentos em P&D elevam os padrões de vida e produtividade no trabalho, invariavelmente causam maiores crescimentos de produtividade e eventualmente ajudarão a economia estadunidense a crescer mais do que 3% em um ano. Neste sentido, fica claro que a estratégia do país Estados Unidos, uma das maiores potências mundiais, envolve investimentos em Pesquisa e Desenvolvimento cada vez maiores, já que este é um dos importantes fatores que guiam a prosperidade financeira do país.

Segundo o Instituto Estatístico da Unesco (2021), cada dólar investido em P&D gera aproximadamente dois dólares em retorno. Mesmo que esta taxa varie, P&D é um importante dirigente do crescimento econômico de qualquer país.

> “Estudos descobriram que cada dólar investido em P&D gera aproximadamente dois dólares de retorno. Embora a taxa varie, P&D é um importante dirigente do crescimento econômico. Para alcançar este potencial, os governos precisam de dados confiáveis e precisos. Em troca, a UIS produz uma vasta quantidade de indicadores em recursos humanos e financeiros investidos em P&D para países em qualquer estágio de desenvolvimento.”

Fonte: Instituto de Estatísticas da Unesco, 2021. Tradução própria.

Informações extraídas da base de dados da Unesco (2021) demonstram que o constante aumento dos investimentos em inovação, pesquisa e desenvolvimento interno é de fato uma tendência mundial em prol da evolução de cada nação, e tem essencial participação na prosperidade financeira de um país e no equilíbrio da desigualdade e da pobreza. Analisando valores históricos de investimentos em P&D feitos por empresas privadas ao redor do mundo entre 2000 e 2018, conclui-se que a variação nesta vertente de aplicação é quase unânime: em média, os países alocam ano a ano, através de suas empresas, porcentagens cada vez maiores de seus PIBs. No ano 2000, a média mundial foi 0,62% do PIB investido em P&D via empresas privadas, e entre os primeiros 5 anos do conjunto de dados (2000-2004), foi de 0,64%. Por sua vez, o ano 2018 apresentou média de 0,76%, e entre os últimos 5 anos (2014-2018), a média foi 0,74%. As médias gerais dos países em relação ao PIB (agora considerando o investimento nacional em P&D, não apenas por parte das empresas privadas) foram de 0,91% em 2000 e 1,08% em 2018, sendo 0,88% em média entre os primeiros 5 anos e 1,04% entre os últimos 5 anos, demonstrando que os países que buscam desenvolvimento contínuo recorrem sistematicamente a investimentos em Pesquisa e Desenvolvimento. Verificando os 6 países que a cada ano entre 2000 e 2018 mais alocaram verbas no financiamento para desenvolvimento de empresas através de políticas públicas, vários se repetem e apenas 9 ao redor do mundo formam a lista: Estados Unidos, Rússia, Alemanha, França, Reino Unido, China, Coreia do Sul, Japão e Espanha. Como é de se esperar, todos figuram entre os países mais desenvolvidos do mundo. O Brasil, por sua vez, mesmo tendo o PIB maior que alguns destes, se mantém praticamente estabilizado nos mesmo níveis desde o início do século.

Tabela 1 – Investimentos totais do Brasil em pesquisa e desenvolvimento ao longo dos anos, como percentual do PIB, de 2000 a 2018.

Investimentos em P&D no Brasil

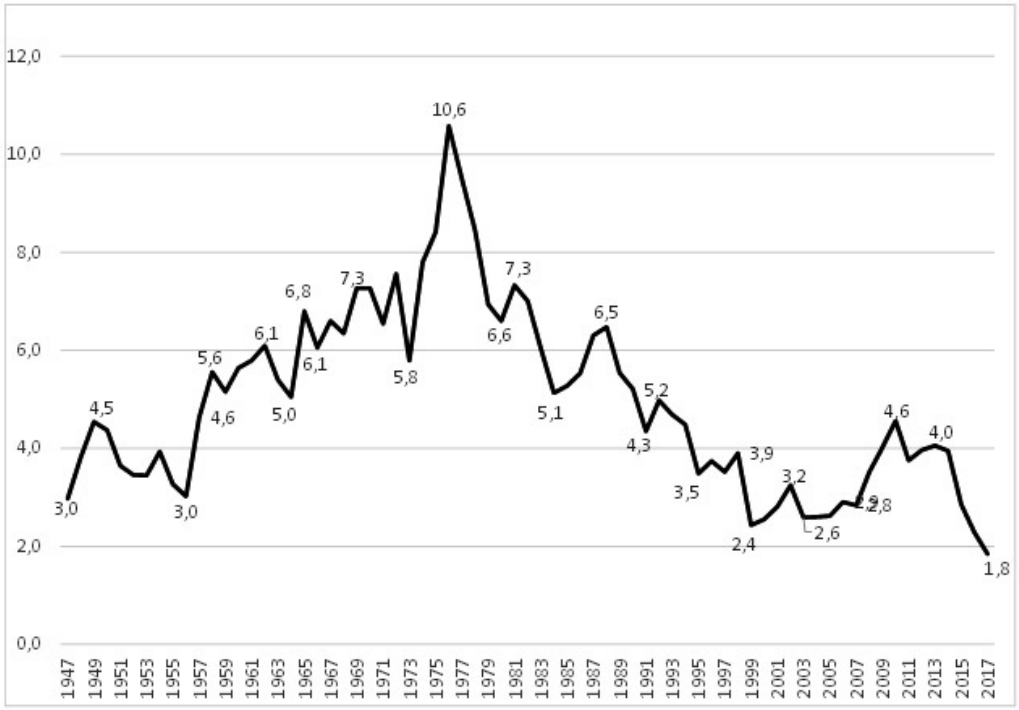
Ano	% do PIB
2000	1.04751
2001	1.06198
2002	1.00968
2003	0.99939
2004	0.96343
2005	1.00246
2006	0.98807
2007	1.08138
2008	1.12904
2009	1.11866
2010	1.15992
2011	1.13966
2012	1.12684
2013	1.19567
2014	1.27131
2015	1.34264
2016	1.26343
2017	1.09392
2018	1.16038

Fonte: Unesco (2021).

Os dispêndios em pesquisa e desenvolvimento no Brasil ocorrem em proporções diferentes das tendências mundiais: conforme demonstrado na tabela acima, no ano 2000, o país tinha o investimento em P&D acima da média global, mas manteve-se

estável e viu ao longo dos anos outros países superarem em muito os valores nacionalmente alocados. Entre 2007 e 2010, a média começou a se erguer, atingindo altas históricas em 2014, 2015 e 2016, mas de 2017 em diante decaiu novamente a valores semelhantes aos do início do século. O Brasil atingiu em 2017 o nível mais baixo de investimentos públicos desde 1947, aponta o Observatório de Política Fiscal Brasileiro (gráfico abaixo). Segundo dados do IBGE, o estado mais inovador de todo o Brasil, São Paulo (FAPESP, 2019), representa 30% do PIB brasileiro e investiu em 2018 o equivalente a 1,24% do seu PIB em P&D. Objetivando tornar-se novamente um país de produção valiosa e de crescimento econômico acentuado, os dados de dispêndios globais (Unesco, 2021) demonstram ser imprescindível e urgente que o Brasil revise os investimentos para elevar o valor alocado em pesquisa, desenvolvimento, estudos, políticas públicas e capacitação da população. Na mesma linha, é importante para as empresas brasileiras que almejam competir globalmente em suas indústrias, que equiparem os investimentos internos em pesquisa e desenvolvimento aos de competidores internacionais. Entretanto, isso provavelmente significa margens de lucro mais baixas que as de empresas globais, dadas as altas taxas tributárias e a escassez de auxílio público que o empreendedorismo brasileiro enfrenta, dificultando este processo comparativamente a outros países.

Gráfico 2 – Histórico do investimento geral de porcentagem do PIB no Brasil.



Fonte: Observatório de Política Fiscal do Instituto Brasileiro de Economia.

2.1 Formulação do problema

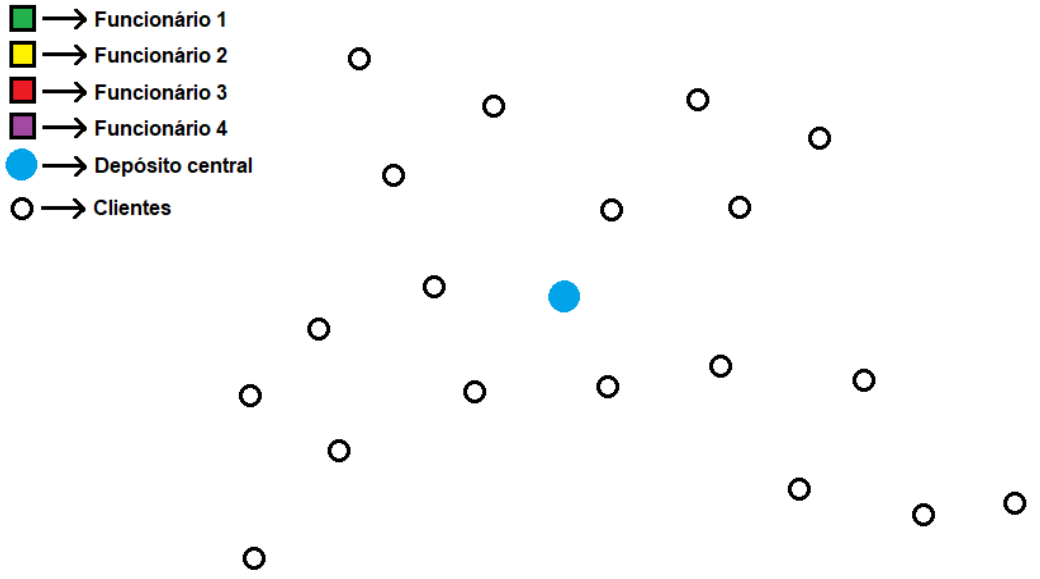
Atualmente, os processos do setor de transportes da empresa-alvo utilizam os funcionários como principal ferramenta intelectual e operacional. Para o procedimento de atendimento, todos os clientes de um turno são planilhados e divididos em grupos setorizados por região, e em seguida atribuídos aos funcionários de transporte para percorrerem os endereços que lhes são designados. Cada funcionário de transporte, responsável por um veículo, cobre os locais na ordem que lhe parece preferencial, buscando terminar seus atendimentos e voltar à empresa dentro de um período de tempo predefinido. Ocorre que neste processo, desde o planilhamento até o percurso, muitas atividades operacionais são feitas à mão e muitas atividades intelectuais são decididas de forma subjetiva, portanto imperfeições ocorrem e a empresa acaba gastando mais do que o necessário. Foi durante um decurso imersivo de voluntariado que o processo foi observado e o gargalo operacional pôde ser identificado, e com isso foi iniciada a busca por soluções. Para tanto, veio à tona a questão: quais são os

ganhos que podem ocorrer ao remodelar esta operação para uma configuração computadorizada?

Compreendeu-se que uma forma computadorizada totalmente automática, utilizando aprendizagem de máquina, poderia executar a tarefa com exatamente os mesmos objetivos, visando otimizar os custos e diminuir riscos: sem o risco de falhar na escrita, confundir, ou efetuar uma previsão muito destoante da realidade. Ainda, diversas destas formas automatizadas foram amplamente estudadas no âmbito da ciência da computação e da otimização combinatória, existindo portanto ferramentas disponíveis online para auxiliar na resolução do problema chamado de PRVJT - Problema de Roteamento de Veículos com Janelas de Tempo (do inglês: *VRPTW - Vehicle Routing Problem with Time Windows*).

A resolução do problema pode ser melhor esclarecida com o auxílio das imagens abaixo. Ilustra-se um conjunto de clientes em locais distintos em torno do depósito central:

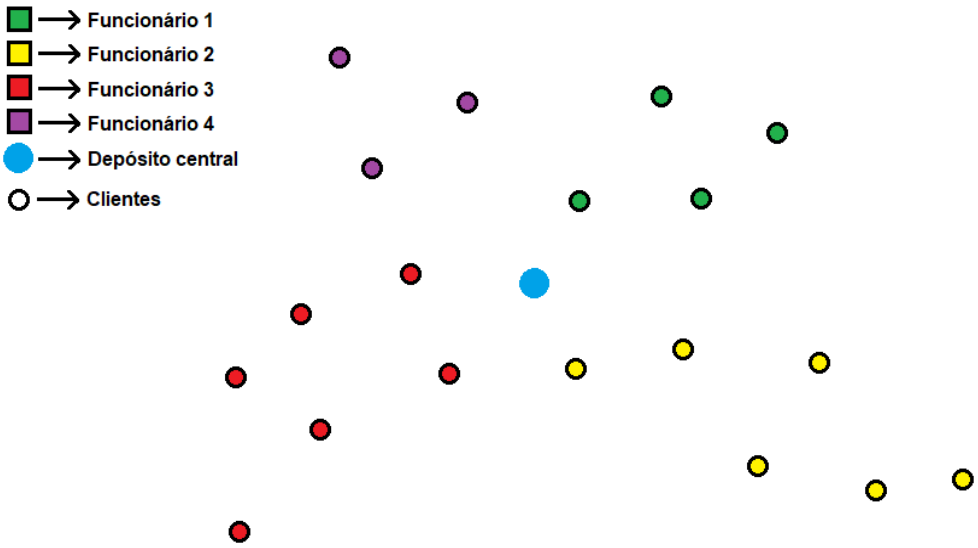
Figura 1 – Demonstração de clientes em localidades aleatórias a serem visitados pelos funcionários.



Fonte: Autoria própria.

Considera-se que todos os clientes devem ser visitados pelos funcionários disponíveis, que ao fim retornarão ao depósito central. Como os clientes devem ser distribuídos entre os funcionários de forma a otimizar a operação? Inicialmente, deve-se efetuar o método de clusterização, que consiste na separação em blocos (melhor abordado na seção *4. Método*), para todos os clientes:

Figura 2 – Demonstração de clientes divididos em blocos, que representam as atribuições de cada funcionário.

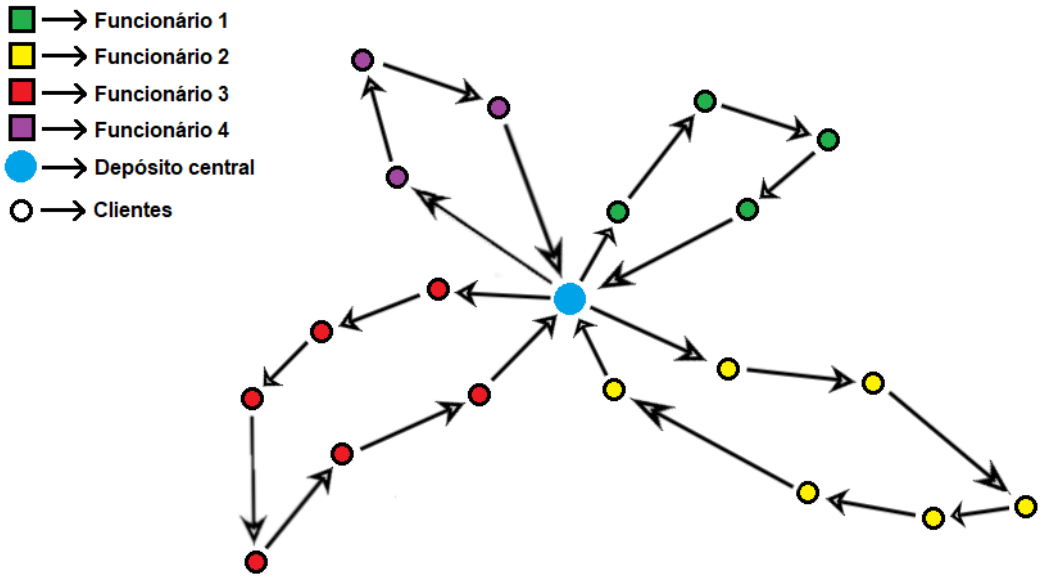


Fonte: Autoria própria.

Com os clientes atribuídos, a segunda questão a ser solucionada é: Em quais ordens cada funcionário deve cobrir seus clientes de forma a otimizar a operação?. Apresentam-se abaixo dois formatos para a resolução deste problema. No primeiro, todos os funcionários saem do depósito central, cobrem todos os clientes e em seguida retornam ao depósito. Chamamos este método de PRV com Centróide Único e Depósito Único. Centróides correspondem aos pontos de referência para a atribuição dos clientes. Um centróide é uma localização para a qual é medida a distância entre ela e o cliente, a fim de atribuir o cliente prioritariamente ao centróide mais próximo. Na execução do programa, que ocorre por várias rodadas, os centróides variam de posição ao fim de cada rodada conforme as posições dos clientes a eles atribuídos, sendo a posição de um centróide em dada rodada igual à média das posições dos

clientes atribuídos a este na rodada anterior. Esta variação, assim como a execução de várias rodadas, tem o objetivo de otimizar a atribuição de cada cliente aos funcionários mais adequados. Ou seja, os centróides aqui são simplesmente as posições variantes dos funcionários. No PRV com Centróide Único e Depósito Único, os trajetos ficam da seguinte forma:

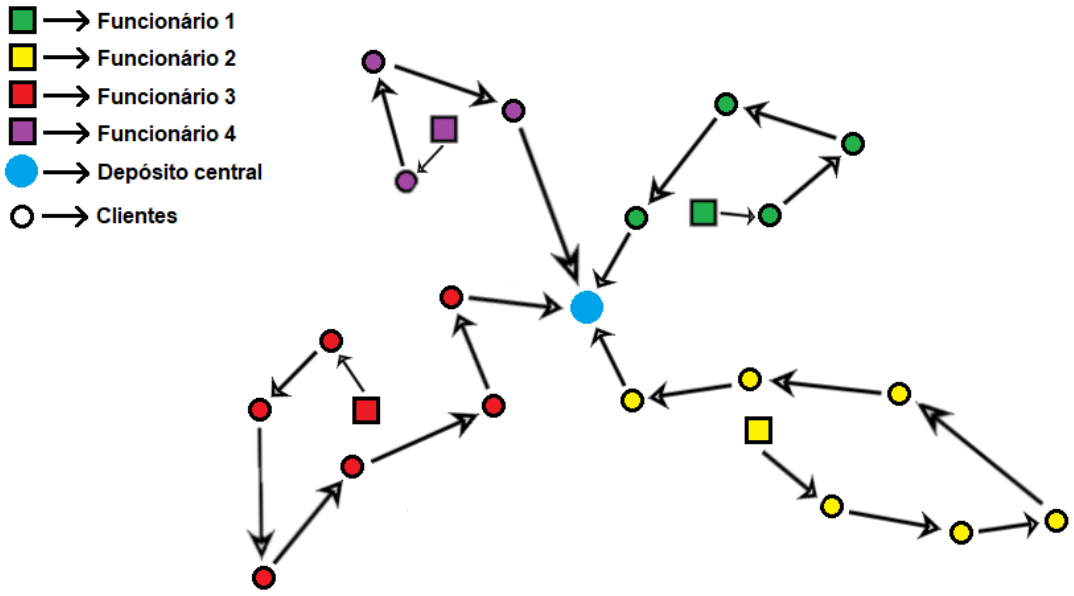
Figura 3 – Rotas com centróides e depósitos únicos.



Fonte: Autoria própria.

No segundo formato, cada funcionário tem um ponto de partida próprio e, ao fim, todos vão ao depósito central. Chamamos esta versão de PRV com Centróides Variados e Depósito Único:

Figura 4 – Rotas com centróides variados e depósito único.



Fonte: Autoria própria.

O software desenvolvido permite estabelecer, para os funcionários da operação, os endereços individuais de origem e de destino. No primeiro formato de divisão, estabelecemos todas as origens e destinos como um endereço único, referente ao local do depósito. No segundo formato, o destino é o mesmo para todos (depósito final), mas as origens são individuais para cada funcionário. Na operação e nos testes do software, utilizamos o formato de origem variável. Esta aplicação é mais econômica, pois não tem os custos de trajeto do depósito até a primeira localização, como ocorre na primeira demonstração. Assim, se for possível para uma empresa, costuma ser preferível para uma operação mais econômica.

.

2.2 Objetivo Geral

A partir do questionamento sobre as diferenças operacionais advindas da roteirização feita por pessoas e da feita por algoritmos, temos como hipótese inicial que uma roteirização executada por um algoritmo pode transformar positivamente a operação da empresa em termos financeiros, de qualidade e de eficiência. O principal objetivo do desenvolvimento deste projeto, portanto, é a remodelagem do processo de

logística urbana a partir da formulação, adequação e avaliação de um software que auxilie a ter e manter a empresa-alvo como referência em logística de excelência, desenvolvimento tecnológico e atendimento ao cliente, através da entrega de meios de execução e avaliação constantes, baratos, claros e assertivos para que as atividades envoltas pelo processo de logística urbana em roteamento de veículos e atendimento aos clientes sejam supridas adequadamente e de forma contínua. Com isso, espera-se uma participação ativa na geração de processos mais hábeis e uma bibliografia nacional mais completa, de forma que outras pessoas possam reproduzir a resolução aqui expressa, em prol de um mundo mais informado, eficiente e desenvolvido.

2.3 Objetivos específicos

São, então, os objetivos específicos do projeto para verificação da hipótese:

- Conhecer o processo exercido atualmente;
- Verificar requisitos para efetuar a transformação do processo;
- Formular um algoritmo para efetuar o roteamento de múltiplos veículos de forma a otimizar a alocação dos clientes e otimizar o ordenamento das localidades percorridas;
- Adequar o aplicativo conforme as especificidades e desejos da empresa;
- Testar o aplicativo até garantir a eficiência em todos os casos;
- Medir e relatar os ganhos.

2.4 Objetivos do Software

Os objetivos buscados pela melhoria de processos no setor de logística de rotas de uma empresa vão desde a eliminação de trabalho manual, passando pela elevação da capacidade operacional, até o aumento do valor percebido pelo cliente à marca. Abaixo, explicitam-se os principais objetivos a serem alcançados pela operação com a implementação do software desenvolvido:

1) As especificidades objetivas propostas por este projeto iniciam-se com a **economia de custos da operação**. O desenvolvimento partiu da intenção de poupar trabalho **na retaguarda do setor**, de forma que fosse possível executar as mesmas atividades de construção de rotas sem a necessidade de utilizar tantos funcionários como a operação original. Eventualmente o software atingiu potencial de economizar toda a carga de trabalho responsável pela construção das rotas, a partir da automação do software, extração direta dos bancos de dados e agendamento diário das execuções.

2) A **economia de recursos humanos** estendeu-se para os funcionários **da linha de frente**, quando a atividade se mostrou alcançável com os mesmos resultados do formato original pelo formato automático, utilizando menos funcionários. Isto foi possível a partir de duas funções do algoritmo: a primeira é a atribuição de clientes priorizando funcionários que já estiverem na operação. A segunda função redistribui os clientes de um funcionário que tenha muito tempo livre aos demais, desde que os custos da operação não aumentem após a redistribuição.

3) Ainda em relação à redução de custos, a operação automatizada objetivou e exerceu uma **redução drástica na quilometragem** rodada por cada funcionário, permitindo a economia de gasolina e manutenção dos veículos. Isto porque ela utiliza métricas precisas para avaliar qual funcionário deve visitar determinado endereço e qual é o melhor caminho para chegar lá, **objetivando reduzir o percurso total e a duração decorrida**.

4) Somando às economias de custos, o **ganho de produtividade** também objetiva a elevação de ganhos financeiros da operação. A partir da diferença comparativa entre as durações das rotas construídas manualmente e automaticamente, e também a partir do acréscimo na disponibilidade de funcionários de campo que foram poupados pelo algoritmo, demonstrou-se que a operação aumentou a sua capacidade produtiva para ofertar mais serviços à demanda existente.

5) O sistema objetiva um **controle completo da operação** e a **capacidade de aprofundar estudos quanto aos melhores formatos operacionais**. Para tal, utiliza-se do acompanhamento das escalas de trabalho a partir de previsões de tempo e distância, desde o momento inicial dos funcionários até o final do expediente. Assim, é possível modelar diversos formatos da operação e testar seus desempenhos.

6) O **controle de qualidade sobre os recursos humanos** também é um elemento chave no sucesso da operação, pois apenas desta forma é possível saber se a empresa atende toda a sua capacidade disponível e se está na hora de expandir a produção. Isto é de grande auxílio na tomada de decisão entre melhorar uma operação existente ou aumentá-la. Um panorama individualizado para os funcionários garante que a equipe de RH saiba quais funcionários são mais produtivos, incluindo quem são os funcionários que mais e menos atendem e os que mais e menos gastam recursos da empresa para executar suas atividades. Com isso, torna-se possível diagnosticar quando um funcionário executa atividades alheias em horário de trabalho ou diariamente não desempenha conforme o esperado, mantendo o controle sobre o produto entregue ao cliente.

7) Um objetivo essencial para uma operação funcional é a **redução de atrasos** por parte da equipe de linha de frente. É importante que todos os horários combinados com os clientes sejam cumpridos, e para isso a operação deve ser meticulosamente mensurada para antes de tudo saber se o funcionário é capaz de entregar o que está sendo combinado, ou se o alcance de um possível combinado está além da capacidade do funcionário. O controle das durações de trajeto é necessário para que a operação esteja sempre dentro dos limites do que um servidor é capaz de entregar.

8) Por fim, a melhoria destes processos tem o objetivo de fazer com que a empresa entregue **mais qualidade ao cliente final**, através da confiança no cumprimento

dos combinados e da disponibilidade de oferta quando o cliente efetuar uma demanda. Este ganho tem um valor imensurável devido à sua subjetividade, mas é possivelmente o objetivo que gera mais retorno ao sucesso da empresa, dentre os abordados neste trabalho.

2.5 Justificativa

A existência dos amplos estudos realizados no âmbito da logística de veículos e a livre disponibilidade dessas informações na internet foram grandes motivadores para o desenvolvimento desta melhoria de processo logístico e do presente documento, através das noções de que i) existem algoritmos auxiliares gratuitos disponíveis, ii) o problema é solucionável utilizando uma heurística, iii) é um problema importante para a computação e para as empresas, iv) é um problema mais esclarecido na bibliografia estrangeira comparada à nacional, e v) muitas empresas brasileiras têm operações de trânsito e demonstram necessitar de intervenções logísticas.

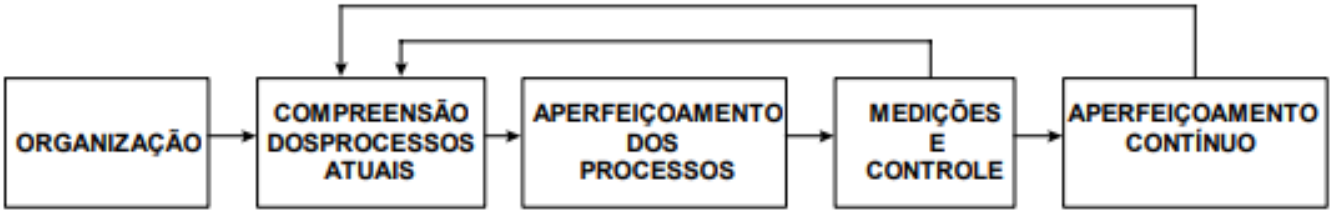
Com o presente artigo, pretende-se divulgar para a bibliografia brasileira e da língua portuguesa os processos e termos por trás das atividades executadas, incrementando a acessibilidade dos brasileiros e demais falantes de português à resolução programática dos temas tratados, a fim de facilitar as pesquisas e tornar possível uma reprodução guiada do problema contemplado.

3. Referencial Teórico

3.1 Remodelagem de processos

A remodelagem de processos é por padrão exercida sobre operações tendo como objetivo efetuar alguma melhoria, seja, por exemplo, redução de custos, aumento da operação, melhoria da capacidade produtiva, resolução de um gargalo ou simplificação do processo. É a atividade de reformular uma operação vigente para o seu formato melhorado. De acordo com Roglio e Selig (1998), “o aperfeiçoamento de processos exige uma revisão da estrutura organizacional da empresa, a partir de uma análise de todas as atividades que compõem cada processo. Este estudo tem como objetivo a definição das atividades que agregam valor e a sua adequação às necessidades da empresa e dos clientes internos e externos”. A remodelagem de processos em sua forma adequada deve seguir 5 fases, formando o ciclo ideal do aperfeiçoamento de processos segundo Harrington (1993):

Figura 5 – Ciclo de 5 fases do aperfeiçoamento de processos



Fonte: Harrington, 1993 apud Roglio e Selig, 1998. Extraída de Roglio e Selig, 1998.

Foi este ciclo proposto por Harrington que foi seguido para a compreensão do formato vigente da operação do setor de logística de rotas e em seguida a reprodução da operação em seu formato algorítmico, como para o aperfeiçoamento da operação do software, até encontrar um formato em que a operação automática superasse a operação original em todas as tentativas, sempre baseando-se em métricas de controle de desempenho. As fases propostas serão usadas continuamente para a melhoria constante da operação algorítmica de forma a aproximar cada vez mais os resultados

encontrados pelo software das soluções ótimas globais (as absolutas melhores resoluções para dado problema).

Observam-se duas distinções quanto aos processos internos, ao tratarmos dessa operação logística. Uma se refere ao processo de roteamento em si, a outra ao processo exercido pelo funcionário. No formato original da operação, o processo do funcionário agregava o processo de roteamento. Com a vigência do algoritmo, uma conclusão possível é que este processo de roteamento se torna mais complexo para envolver mais parâmetros lógicos que visam a exatidão, por outro lado ele saiu do escopo do funcionário e a atividade do operador se tornou mais simples. Portanto, ao tratarmos de processos corporativos como as atividades a serem executadas pelos funcionários, pode-se considerar que o processo como um todo se tornou mais simples e eficiente.

3.1.1 Reengenharia

Para Hammer e Champy (1993 apud Souza, 2016), a reengenharia se pauta no “pensamento descontinuado”. Engloba a readaptação de processos corporativos, envolvendo estruturas organizacionais, sistemas de informação e valores da organização, objetivando mudanças de direção significativas nos resultados dos negócios da organização. Dentre outras características, a reengenharia também busca a fusão de tarefas e a eliminação do desnecessário.

3.1.2 Gestão de Processos

A utilização de indicadores para a manutenção de processos corporativos, e a avaliação detalhada de cada etapa da atividade, são partes da composição da Gestão de Processos. A pesquisa operacional exercida para averiguação de desempenho do software, referente à utilização da estatística na tomada de decisão, está portanto vinculada à vertente da administração voltada à gestão dos processos, por mais que o termo seja frequentemente abordado nos campos matemático e computacional. Segundo Mello et al. (2002 apud Souza, 2016), a gestão de processos proporciona a definição da maneira mais

adequada para a efetivação de cada atividade do processo em si e de indicadores gerenciais que permitirão a medição, análise e melhoria de cada processo.

3.1.3 Gestão da Qualidade

Deming (1990) aborda a gestão de processos em uma ótica voltada a técnicas e ferramentas estatísticas para controle de qualidade. Em seu modelo, que foi chamado de Método de Gerenciamento (*Management Method*), Deming aprimorou a ideia de Shewhart (1939) que dá origem ao conhecido Ciclo PDCA (*plan, do, check, act*), conceito que envolve as fases cíclicas planejar, executar, verificar e agir. Deming aloca uma importância superior sobre a gestão da qualidade, utilizando métodos baseados em informações para efetuar decisões, e ferramentas para auxiliar suas análises.

3.2 PCV - Problema do Caixeiro Viajante

O PCV (do inglês *TSP - Travelling Salesman Problem*) é considerado o mais conhecido problema da computação em otimização combinatória. Seu objetivo é simples: encontrar o caminho mais curto dentre uma lista de localidades. A resolução do PCV objetiva minimizar para um único veículo a distância percorrida entre um conjunto de endereços (Papadimitriou e Steiglitz, 1982). A ferramenta escolhida para resolver o PCV chama-se OSRM - Open Source Routing Machine, que pode ser acessada no software aberto disponibilizado por este projeto. Apesar de descomplicado, a execução do problema se torna bastante trabalhosa para um computador conforme se aloca uma quantidade alta de localidades a serem otimizadas. Por este motivo utiliza-se uma heurística, que é uma resolução rapidamente solucionada se aproximando da melhor forma possível.

No projeto, o Problema do Caixeiro Viajante é utilizado como otimizador de trajetos dos funcionários em uma escala. Para isso, após atribuídos todos os clientes da escala, o PCV é resolvido para cada funcionário, a fim de encontrar para cada um

deles a melhor ordem de visita dentre as localizações dos clientes que lhes são atribuídos.

3.3 PRV - Problema de Roteamento de Veículos

O PRV (do inglês *VRP - Vehicle Routing Problem*), proposto por Dantzig e Ramser (1959), envolve um conjunto de repetições do Problema do Caixeiro Viajante. Utiliza o PCV em sua composição porém para múltiplos veículos envolvidos. O PRV busca otimizar não apenas a ordem das localidades de cada automóvel, mas também a distribuição das localidades dentre os veículos disponíveis. Esta distribuição das localidades se aproxima do formato ótimo através de métodos de clusterização. É através de uma variação do PRV que o software desenvolvido propõe melhorar o processo corporativo, com implementação do formato automático de resolução.

3.3.1 PRVJT - Problema de Roteamento de Veículos com Janelas de Tempo

O PRVJT (do inglês: *VRPTW - Vehicle Routing Problem with Time Windows*) é uma das variações do problema PRV. Envolve restrições de horários e tempos de duração, fator que influencia na capacidade de atendimento de cada veículo e portanto na otimização dos resultados. É esta a variação do PRV utilizada na versão final do software descrito neste projeto e implementado à empresa.

3.3.2 PRVC - Problema de Roteamento de Veículos Capacitados

PRVC (do inglês: *CVRP - Capacitated Vehicle Routing Problem*) é uma variação do PRV que envolve limites de capacidade para os veículos envolvidos. A partir desta funcionalidade, é possível atribuir regras para limitar a quantidade de determinado objeto que o veículo é capaz de suportar ou limitar o conteúdo material disponível no veículo para efetuar a operação.

3.4 Clusterização

A clusterização (Caimi, 1993) é a técnica de particionamento de dados, muito importante para a resolução do problema de roteamento de múltiplos veículos na etapa de atribuição dos clientes aos funcionários. A qualidade da resolução depende diretamente das técnicas de clusterização utilizadas, pois são essas que definem como as alocações dos clientes são feitas e como consequência as distâncias e durações totais de trajeto. O desenvolvimento até alcançar um particionamento satisfatório iniciou-se com o método chamado de modelo gravitacional, onde os clientes eram entregues aos funcionários de forma a simular um plano gravitacional: os clientes recaíam sempre sobre os funcionários mais próximos. Convencionou-se chamar este método de "método de divisão estática", pois as referências de localização dos funcionários permaneciam as mesmas até o fim do processo.

O modelo gravitacional funciona, mas por si só passou a falhar eventualmente, quando as restrições de tempo foram incluídas no processo. Devido a essas restrições, os funcionários mais próximos a um cliente se tornavam indisponíveis e grandes locomoções tinham que ocorrer para que o cliente pudesse ser atendido por outro funcionário. Para solucionar isso, o método de clusterização por k-médias foi incluído, com o objetivo de fazer as inevitáveis locomoções serem minimizadas. Com este método, as referências de localização para os funcionários passaram a mudar de rodada em rodada, com base no ponto médio entre todas as localizações até então absorvidas pelo funcionário, fazendo então com que os grupos ficassem divididos de maneira mais natural, com menos contraste entre eles. Ao mesmo tempo, o método permitiu um equilíbrio melhor na quantidade de atribuições de cada funcionário.

Mesmo no modelo com coordenadas dinamicamente variantes, as partições separadas visivelmente ainda podiam ser melhoradas. Os grupos definidos ainda estavam com formatos bem demarcados, em uma geometria próxima a um círculo. Assim, era observável que o tempo ainda podia ser economizado ao atribuir locais no decorrer de um trajeto a um funcionário que invariavelmente passaria por lá. A solução proposta a reduzir essa geometria e melhorar o tempo gasto atingiu o objetivo devido à

anulação da locomoção de um funcionário próximo a um cliente, caso outro funcionário já fosse passar por aquele ponto. Dessa forma, o segundo funcionário só gastaria o tempo de parada, dispensando o tempo de deslocamento que o primeiro faria. Isso foi alcançado misturando o método de k-médias com o primeiro método citado, o modelo gravitacional. Mas os pontos gravitacionais não seriam mais a localização de um funcionário, e sim os pontos do trajeto que o funcionário faria desde o momento inicial da operação até a ida para o depósito.

Uma clusterização eficaz para todos os testes, portanto, foi atingida com a mescla do método gravitacional com o método k-médias.

3.5 OSM - Open Street Map

OSM é uma base de dados pública que compila informações de rotas veiculares ao redor do mundo. A partir dela é possível buscar um endereço, verificar velocidade de qualquer via, ou visualizar mapas e trajetos. Os mapas presentes no projeto, por exemplo na figura 7, são construídos por uma biblioteca Python através de dados da Open Street Map.

3.6 OSRM - Open Source Routing Machine

Open Source Routing Machine (na tradução literal: Máquina de Roteamento de Código Aberto), comumente chamado de OSRM, é um mecanismo de código aberto para otimização de um conjunto de endereços, resolução do PCV. Ele combina dados do Open Street Map com algoritmos de roteamento, e a partir do mecanismo é possível otimizar repetidamente os trajetos de veículos, auxiliando na resolução do Problema de Roteamento de Veículos. É construído na linguagem de programação C++, mas outros mecanismos tornam possível executá-lo via Python. O presente projeto utilizou o software de virtualização Docker para executar a implementação original do OSRM no computador e acessá-lo via requisição web local. Em aplicação neste projeto, o mecanismo OSRM foi utilizado tanto para otimizar as ordens entre os trajetos dos funcionários, como para verificar as distâncias entre um ponto e outro na etapa de clusterização para confecção dos trajetos.

Na otimização de trajetos com menos de dez localidades, a OSRM utiliza o método força bruta, que consiste na testagem de todas as possibilidades. Portanto, para menos de 10 localidades a ferramenta retorna o formato absolutamente ótimo. A OSRM torna isso possível a partir da verificação entre todos os possíveis trajetos que percorrem as localidades, e a escolha do que tiver a distância total mais curta. Para trajetos com 10 ou mais localidades, o método utilizado é chamado de Algoritmo de Inserção Mais Distante (do inglês: *farthest-insertion algorithm*), método considerado uma heurística gananciosa (do inglês: *greedy heuristic*).

O Algoritmo de Inserção Mais Distante é um método que consiste na construção de um trajeto utilizando as distâncias euclidianas entre os pontos. O processo inicia-se verificando o local mais distante do ponto inicial. Em seguida, verifica-se o outro ponto mais distante a este ponto encontrado, que intermedeia o trajeto então existente (entre os dois pontos atribuídos até agora) de forma a minimizar sua distância, e assim por diante até passar por todos os pontos locais atribuindo-os como intermediários ao trajeto (ou linha) mais próximo. O resultado final é uma polilinha (um “polígono aberto”) onde cada localidade (tirando a inicial e a final) tem duas ligações. Esse processo é considerado uma heurística gananciosa pois aborda apenas as otimizações imediatas, sem comparar todos com todos.

A heurística gananciosa é um tipo de resolução que a cada iteração seleciona a decisão que resulte em maior melhoria do valor objetivo durante a iteração atual (Hart & Shogan, 1987). Ou seja, o algoritmo ganancioso segue o objetivo de selecionar a escolha ótima local em cada estágio do algoritmo, em detrimento à escolha ótima global, que seria um processo mais difícil e mais demorado. Por esta razão, ela usualmente não produz o resultado definitivamente ótimo, mas se aproxima desse resultado em um tempo curto de execução.

3.7 Docker

Docker é uma plataforma de código aberto utilizada para automatizar a implantação de aplicativos em sistemas. A plataforma permite que o mecanismo OSRM seja executado através dela, tornando-o disponível em um servidor local na máquina.

Para tal, o Docker simula outro sistema operacional dentro do computador e torna o mecanismo OSRM disponível através dele, que passa a ser acessível pelo servidor local. Por exemplo, sempre que executado, o OSRM se torna disponível para requisições http através do link http://localhost/. Portanto para saber a distância entre o ponto 1 e o ponto 2, deve-se acessar http://localhost:8080/trip/v1/driving/coordenada_1;coordenada_2 e ler os resultados através do Python.

Consideremos a distância e duração entre “UnB, Brasília, DF” (-15.75799565, -47.8713539317) e “Aeroporto Internacional de Brasília, DF” (-15.87177275, -47.9140875): a solicitação é enviada através de http://localhost:8080/trip/v1/driving/-47.8713539317,-15.75799565;-47.9140875,-15.87177275, e o resultado da requisição é retornado no chamado formato JSON da seguinte forma:

{"code":"Ok","trips":[{"geometry":"rvd_Bp{dcHwAyK|c@e[~CoKvdAjLbx@b{@|[wFpb@fOqM`v@zH`@~BuI~]pJhdA|t@lcDzyEjRfM`rDj`@bJuNeA_q@~WkYcC{}@HrMr@bs@_IvHqFsLeMRiEzKrLhLr@`q@qHhLydAeQkkBiM_RwLeoCmdEg{@mx@gl@iX{|@yQagB~BkBySiUdD@sR{W|EgCkMhLeDgQ`D}AgJeO\\YuH","legs":[{"steps":[],"distance":20606.7,"duration":1436,"summary":"","weight":1436},{"steps":[],"distance":20022.9,"duration":1376.3,"summary":"","weight":1376.3}],"distance":40629.6,"duration":2812.3,"weight_name":"routability","weight":2812.3}],"waypoints":[{"waypoint_index":0,"trips_index":0,"location":[-47.871452,-15.758021],"name":"","distance":10.859409,"hint":"TkYBgCDrG4BCAAAAVwAAAA8AAAAAAAAAdP_aQfBREEKghMNAAAAAAEIAAABXAAAADwAAAAAAACiAAAAJIol_TuND_-GiiX9VI0P_wEAXwp2B1CK"},{"waypoint_index":1,"trips_index":0,"location":[-47.913949,-15.873726],"name":"","distance":216.614137,"hint":"vO4cgP___3_hAAAAVwEAAAAAAAaAAAAeI56Q-FIA0MAAAAACWLnQeEAAABXAQAAAAAAABoAAACiAAAAI-Qk_ULJDf-Y4yT949AN_wAArxB2B1CK"}]}

Em seguida, basta extrair os valores de “distance” e “duration” (distância e duração) e dividi-los respectivamente por 1000 e 60, para transferir a distância de metros para quilômetros e a duração de segundos para minutos. Assim se tem o tamanho do percurso e o tempo gasto entre a UnB e o Aeroporto de Brasília.

3.8 Python e suas ferramentas

Python é o nome da linguagem de programação utilizada para desenvolver o algoritmo resolutivo do presente projeto. É uma das linguagens mais utilizadas no

mundo e também a que mais cresceu nos últimos anos (TIOBE Index, 2021), devido à propagação de ferramentas construídas pelos seus usuários. Isso faz com que o Python tenha um desempenho especialmente alto em acessibilidade a funcionalidades, permitindo que o usuário utilize ferramentas pré-desenvolvidas gratuitamente, úteis para resolver problemas cotidianos processuais e matemáticos. As ferramentas de outros usuários podem ser utilizadas a partir de duas principais formas: copiando ou baixando o arquivo de texto contendo o algoritmo desejado (por exemplo o código aberto disponibilizado por este projeto, acessível em https://github.com/victorccaldas/tsp_pcv_vrotas/), ou a partir de uma biblioteca, que é um conjunto de arquivos de código integralizados pela plataforma oficial do Python PyPI. As bibliotecas são normalmente gratuitas.

A confecção do software para roteamento de múltiplos veículos utilizou algumas bibliotecas principais, brevemente resumidas a seguir:

3.8.1 Tkinter

Esta é uma biblioteca focada em desenvolvimento de interfaces visuais, comumente chamadas de GUI (*Graphical User Interface:* Interface Gráfica do Usuário). A partir dela, foi construído todo o *front-end* (a frente interativa) do aplicativo. Portanto, todos os botões, textos e entradas visualmente disponíveis no aplicativo foram inseridos pelo Tkinter.

3.8.2 Requests

Requests é uma funcionalidade que permite acessar e extrair dados de endereços web a partir do Python. Por exemplo, caso se objetivasse compilar informações sobre a frequência do Covid-19 nos países, poderia se utilizar *requests* em algum site que tem essas informações e obter os números sem precisar acessar o navegador. A utilidade da função no presente projeto está em extrair do servidor local as informações sobre os trajetos, a partir dos dados pré-calculados pela OSRM. Relembrando, o OSRM pôde ser ativado a partir do Docker, disponibilizando a ferramenta em um servidor local. Assim, a ferramenta se tornou acessível a partir do “próprio endereço web” localhost (servidor local), acessado pelo comando Python:

requests.get(url=http://localhost/)

3.8.3 Pandas

É uma biblioteca para tratar dados em forma de matriz a partir do Python. É a escolha principal dentre os usuários da linguagem para manipulação de tabelas, devido à grande disponibilidade de funcionalidades que a biblioteca dispõe. Através dela, operar dados se torna bem mais rápido e eficaz que no Excel, por exemplo. No projeto, é utilizada em todo agrupamento de informações, tendo como exemplos a compilação dos dados de clientes ou o conjunto de endereços atribuídos a cada funcionário da operação.

3.8.4 Gspread

A biblioteca gspread permite a integração do algoritmo com o Google Drive, mais precisamente o Google Sheets. A partir dela, os conjuntos de dados compilados pelo Pandas podem ser facilmente salvos na nuvem, aumentando o acesso e garantindo a segurança das informações. Isso permite que seja evitada uma etapa intermediária de salvar as tabelas no computador e em seguida colocá-las manualmente no drive.

3.8.5 Geopy

Geopy é uma biblioteca de funcionalidades espaciais, a partir de dados do mapa geográfico. A função mais importante para o projeto é o Nominatim geocoding, que permite a capacidade de obter pontos no mapa em formato de coordenadas através de um endereço em texto. O Google dispõe de uma função semelhante, também utilizada no código para amplificar os resultados encontrados. Outra função importante do Geopy para o desenvolver do algoritmo é obter a distância geodésica entre dois pontos. A distância geodésica (Karney, 2013) é a distância mais curta entre dois pontos no modelo da Terra elipsoidal. Foi bastante utilizada nas versões iniciais do programa, mas por apresentar distâncias em linha reta foi substituída pela distância de trajeto disponível no OSRM, objetivando aumentar a acurácia dos resultados.

3.8.6 Folium e gmplot

Folium e gmplot são bibliotecas utilizadas para plotar locais no mapa e facilitar a visualização. Folium utiliza o mapa de código aberto obtido no Open Street Map (verifique Figura 7), enquanto o gmplot utiliza o mapa do Google Maps (verifique Figura 14). Ambas são utilizadas no programa devido às capacidades individuais de cada uma.

3.8.7 webbrowser

A biblioteca webbrowser permite que através do Python se abra um endereço em algum navegador web (Firefox, Chrome, Explorer, etc.). Os softwares construídos neste projeto utilizam dessa funcionalidade para, a partir do clique dos botões, abrir os mapas gerados em arquivo html pela folium ou gmplot.

3.8.8 Statistics e math

Statistics e math são duas bibliotecas que contemplam variadas operações estatísticas, prontas para serem usadas rapidamente. Funcionam como atalhos para não precisar escrever manualmente determinada operação lógica em uma conta matemática. Por exemplo a função de média entre um conjunto de números: enquanto manualmente seria necessário somar os números do conjunto e dividir a soma pelo tamanho do conjunto (quantidade de números envolvidos), com statistics é possível simplesmente efetuar o comando ‘média(conjunto_de_números)’ e obter o resultado desejado. Math e statistics ajudam a poupar tempo e esforço quando já se sabe qual operação matemática precisa ser executada para obter determinado resultado.

3.8.9 Numpy

Numpy é uma biblioteca do Python para tratamento de números e matrizes. É necessário em algumas utilizações do Pandas para normalizar e padronizar formatos de dados, já que o Pandas muitas vezes depende do Numpy. Numpy estabelece a estrutura básica do Python para manipulação de conjuntos de dados multi-dimensionais, enquanto o Pandas facilita essa manipulação.

Estas são as principais bibliotecas Python utilizadas neste projeto, que são nada mais que ferramentas entregues em pedaços de código e acessíveis por quem tiver interesse. Ferramentas como estas existem para problemas de todo tipo, simples e complexos, e costumam ser encontradas facilmente na internet ao buscar em uma fonte confiável.

4. Método

Este projeto constitui uma melhoria de processo efetuada dentro de uma empresa de grande porte, baseada na transformação da operação de logística urbana efetuada por funcionários para um formato sem a necessidade de intervenção humana. O procedimento para reestruturação do processo manual para o processo automático foi executado a partir da linguagem de programação Python. Além do software, os dados necessários para a execução do programa foram mantidos em planilhas do Google Sheets em forma de bancos de dados. Por estar na nuvem de forma gratuita, a base de dados neste formato foi cômoda neste caso por ser simples e de fácil extração através do Python. Para projetos que requerem bases muito extensas ou uma frequência alta de requisições, o Sheets não seria a melhor escolha pois seu mecanismo limita a quantidade de solicitações por minuto feitas pelo usuário e também a quantidade máxima de células disponíveis por documento.

A otimização desta operação é pautada em três principais melhorias de processos, que são:

1. Otimização do agrupamento de clientes, onde cada grupo é atribuído a um funcionário com um veículo;
2. Otimização da ordenação dos clientes atribuídos a cada veículo, para que o funcionário percorra todas as localidades na configuração mais econômica e rápida;
3. Interface para manutenção e controle dos resultados, para que a equipe da empresa efetue decisões com mais propriedade.

Dentre as três listadas acima, as duas primeiras constituem o problema PRVJT, que é a resolução da roteirização urbana com múltiplos veículos, enquanto a terceira auxilia na observação da otimização efetuada pelo software. A resolução utilizou métodos que envolveram:

A. Uma variação da clusterização por k-médias (*k-means*): k-médias (Duda & Hart, 1973) consiste numa técnica de particionamento de dados

através da aprendizagem não supervisionada de máquina (do inglês: *unsupervised machine learning*). Objetiva o agrupamento por semelhança entre um conjunto de dados, buscando maximizar a similaridade entre os componentes de cada grupo e minimizar similaridades entre componentes entre-grupos. No nosso caso, o objetivo é a separação de blocos de endereços para os quais são enviados cada um dos funcionários de transporte. É uma variação do método k-médias pois outras variantes além do k-médias puro influenciam na composição dos grupos, como por exemplo as localidades do trajeto que o funcionário percorre.

B. Aprendizagem não supervisionada de máquina: a aprendizagem não supervisionada (James et al., 2013) diz respeito à capacidade do algoritmo em reconhecer algum aspecto próprio durante a execução. É quando o usuário ou programador não precisa indicar estes aspectos para que possam ser reconhecidos. No caso deste projeto, se refere à capacidade do código de formar grupos de dados por conta própria a partir das semelhanças e diferenças entre os componentes (as localizações geográficas dos clientes), de forma que ninguém precise designar, tanto quanto à capacidade do código de aprimorar a formação desses grupos conforme o passar das rodadas.

C. Atribuição por modelo gravitacional: no modelo gravitacional (Hatamlou et al., 2012), em lógica semelhante ao k-médias, os grupos de clientes também são influenciados caso algum ponto do trajeto de um funcionário esteja próximo a um cliente ainda não atribuído. Neste caso, este cliente entra no conjunto desse funcionário ─ assim como funciona a gravidade, o modelo utiliza da proximidade à linha do trajeto como critério para a atribuição de endereços. São temporariamente removidos do “plano gravitacional” os centróides que esgotarem a capacidade de tempo.

D. Iteração priorizada: foi encontrado que melhores resultados são alcançados ao organizar os clientes por ordem de prioridade (mais distantes primeiro) antes do particionamento de dados iterativo. A atribuição dos clientes mais distantes primeiro faz com que sejam visitados necessariamente pelos funcionários absolutamente mais próximos a eles, visto que neste momento os funcionários ainda terão disponibilidades altas de tempo; caso contrário, poderia ocorrer de um funcionário mais próximo ficar indisponível por excesso de duração, tornando necessário que outro precisasse se deslocar por uma distância maior e mais demorada para visitar aquele endereço.

E. Resolução do Problema do Caixeiro Viajante: a ferramenta utilizada para otimizar trajetos e mensurar tempos e distâncias das rotas individuais dos funcionários, como proposta do PCV, foi a OSRM, ferramenta que faz cálculos de trajetos utilizando-se do banco de dados público OpenStreetMap. Um estudo (Ilia, 2016) comparou os resultados do OSRM e de outras ferramentas com o mesmo propósito, utilizando 5 milhões de localidades na França. O referencial utilizado para o *benchmark* foi a ferramenta de roteamento do Google, possivelmente por entender que eles têm a maior base para dados de trajetos e trânsito em tempo real, o que significa informações mais precisas, apesar de pagas. As correlações do OSRM com Google foram de 0.95 para distância e 0.92 para tempos de trajeto, resultando que os trajetos no Google são levemente mais longos e mais duráveis que no OSRM. De todas as comparativas do estudo, a OSRM foi a ferramenta com menor taxa de erro quanto ao Google, e portanto foi a escolhida para ser utilizada no projeto.

É importante reafirmar que esta resolução de otimização é uma heurística, que significa que os resultados se aproximam da forma realmente ótima. Este é um problema computacional classificado como NP-difícil, que por definição é pelo menos

tão difícil quanto os problemas mais difíceis em tempo polinomial não-determinístico (NP). Isto quer dizer que não há tempo previsível para que um computador consiga resolver o problema devido à quantidade exponencial de possibilidades a serem testadas conforme se aumenta a quantidade de endereços, podendo demorar meses ou anos para terminar. Portanto, utiliza-se uma heurística para obtermos um resultado ao menos próximo de ótimo em um tempo de solução razoavelmente curto.

A demonstração a seguir expõe a equação matemática do método de clusterização k-médias (Duda & Hart, 1973). O problema envolve um conjunto de pontos x_i em que $i = 1, \ldots, n$ tal que cada x_i do conjunto seja uma localidade de cliente. O retorno da função é um conjunto de k clusters $C_1, \ldots, C_k$, e μ_j são os respectivos centróides. Dessa maneira, a função objetivo é definida como a distância euclidiana entre o i-ésimo ponto e seu centróide. O modelo pode ser definido como:

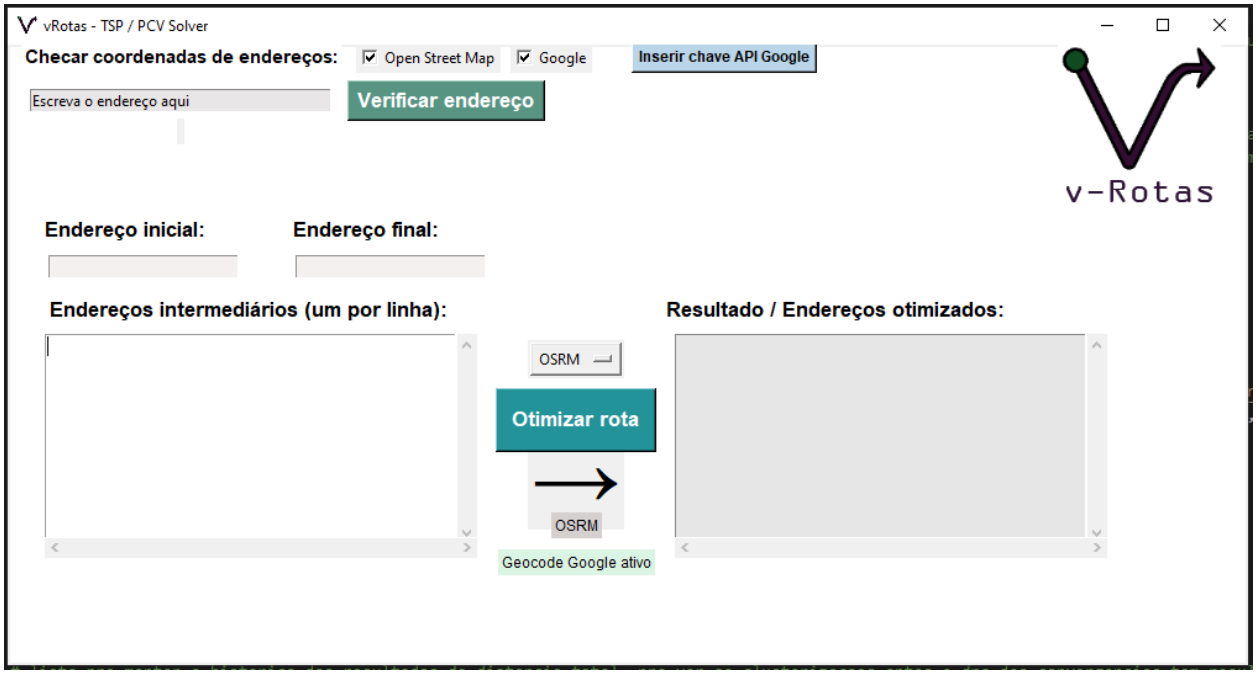
$$\sum_{j=1}^{k} \sum_{i \in C_j} \|x_i - \mu_j\|^2$$

Em que k representa a quantidade de funcionários (veículos) disponíveis. Desta maneira, para 5 veículos, o problema se torna 5-médias e envolve o particionamento de dados em até 5 grupos distintos, formados com base na média entre os componentes de cada grupo.

5. Desenvolvimento e modelagem

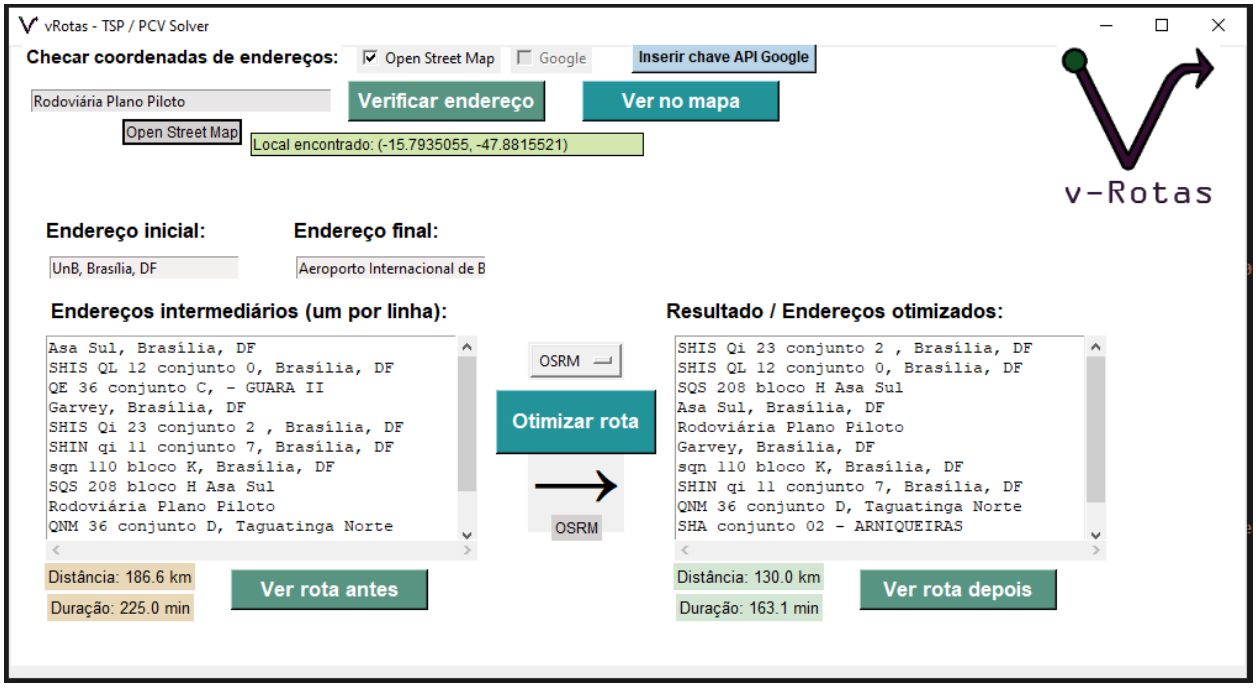
As versões iniciais do software consistiram em uma janela com duas ferramentas: uma para encontrar coordenadas através de endereços, e outra para obter a melhor ordem, o tempo e a distância entre um conjunto de dados endereços (resolução do PCV / TSP). Este aplicativo descrito, com algumas adaptações desde as primeiras versões, é oferecido pelo presente projeto como utilitário de código aberto, disponível no link https://github.com/victorccaldas/tsp_pcv_vrotas/. Foi construído inicialmente com os recursos de desenvolvimento pagos do Google, que concede crédito mensal de utilização, mas com o aumento das execuções para testes e prevendo que em produção diária geraria custos desnecessários para a empresa, os recursos utilizados foram substituídos pelas ferramentas OSRM de código aberto. Ainda é possível ativar o serviço Google para encontrar (geolocalizar) locais mais facilmente, basta inserir uma chave de acesso com crédito disponível na conta (este processo é descrito detalhadamente no link disponível). Esta versão do software é útil para otimizar o trajeto automotivo entre um conjunto de endereços e saber a distância e duração média, visualizar no mapa os trajetos, efetuar comparativos entre uma combinação e outra, encontrar as coordenadas geográficas de um endereço no mapa e visualizar no mapa a localização de determinado endereço.

Figura 6 – Interface do Resolvedor do Problema do Caixeiro Viajante



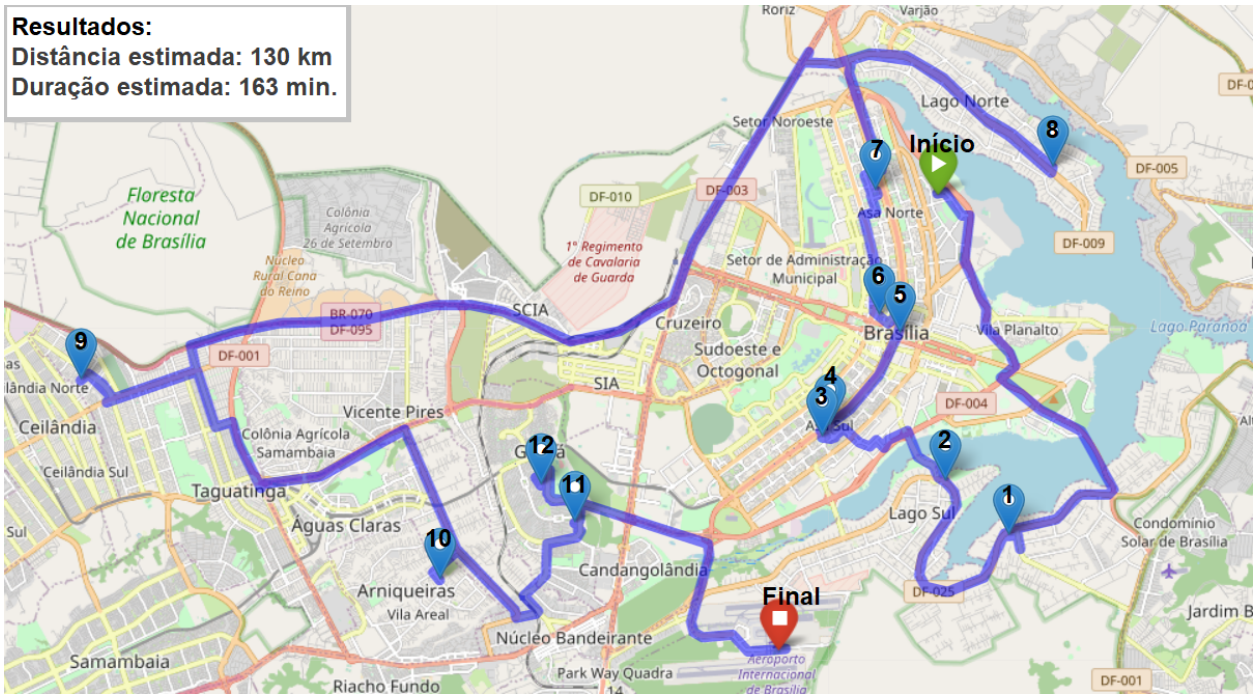
Fonte: Autoria própria. Disponível para download em https://github.com/victorccaldas/tsp_pcv_vrotas

Figura 7 – Operação demonstrada do software TSP / PCV Solver, com endereços intermediários aleatórios, iniciando na Universidade de Brasília e terminando no Aeroporto de Brasília.



Fonte: Autoria própria.

Figura 8 – Demonstração do mapa com o trajeto otimizado, com endereços intermediários aleatórios, iniciando na Universidade de Brasília e terminando no Aeroporto de Brasília.

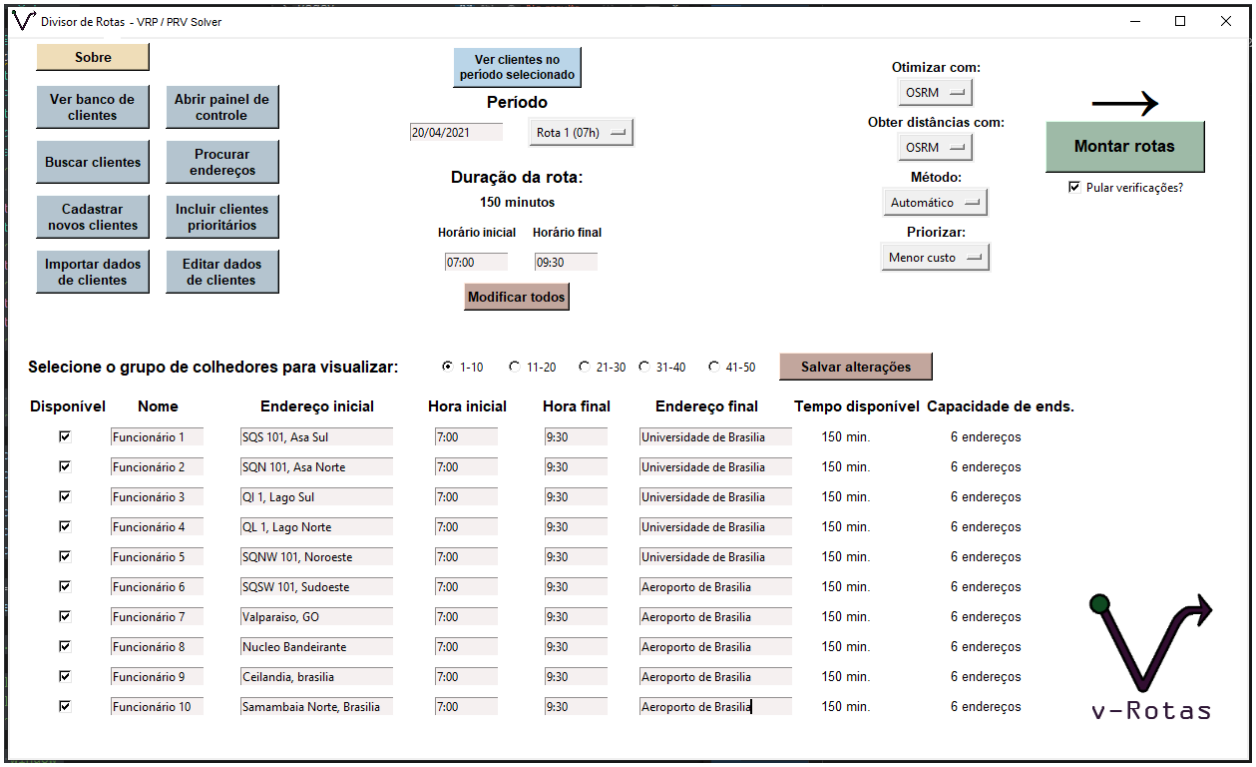


Fonte: Autoria própria.

As versões mais recentes consistem no resolvedor completo do problema de roteamento de veículos, que antes de otimizar os endereços inseridos, cria os blocos que são otimizados. Pela interface, é possível acessar e modificar as informações das bases de dados, definir variáveis e personalizar o problema. Antes de a ferramenta de código aberto ser implantada, algumas tentativas bem sucedidas de otimizar os custos por solicitação ao serviço Directions API da plataforma Google Cloud (função substituída pela OSRM) foram exercidas: uma delas foi medir os trajetos apenas ao fim de cada rodada e estabelecer um limite dinâmico de endereços por veículo, que variava a cada rodada baseando-se em se o limite de tempo havia sido extrapolado ou não (decrescia o limite de endereços em 1 sempre que extrapolasse, ao contrário aumentava 1). Assim, a execução só terminaria quando todos os veículos estivessem dentro do tempo limite - o que causava uma execução mais demorada que a versão atual. Este “limite de capacidade” pode ser modificado para outras utilidades, como estabelecer ao veículo um limite máximo de peso ou definir uma quantidade limitada de

etiquetas disponíveis, por exemplo. A variação do PRV com capacidades chama-se PRVC - Problema de Roteamento de Veículos Capacitados (no inglês, *CVRP*). Assim, este problema completo com tempos limitados denominava-se PRVCJT (*CVRPTW*) - Problema de Roteamento de Veículos Capacitados com Janelas de Tempo.

Figura 9 – Resolvedor do Problema de Roteamento de Veículos Capacitados com Janelas de Tempo, produto principal do projeto.



Fonte: Autoria própria.

Antes de alcançar uma versão final adequada a todos os cenários disponíveis e efetuar sua mensuração operacional para comprovação da eficiência, o processo de construção do algoritmo se pautou na observação e melhoria a partir de tentativas empíricas com variados métodos resolutivos que buscavam o mesmo fim. Dentre as tentativas avaliadas, destaca-se a busca pelo método mais eficiente de clusterização (separação dos grupos de localidades). Observou-se grande melhoria operacional ao comparar a clusterização com centróides dinâmicos versus a clusterização com centróides estáticos: no “método estático”, que caracteriza a versão inicial do

resolvedor com múltiplos veículos, os endereços foram distribuídos baseando-se apenas na posição inicial de cada veículo, de forma que fossem atribuídos a cada funcionário os clientes mais próximos de sua posição de partida. Este formato se provou pelo menos 10% mais eficiente que a operação realista, a partir da comparativa com o formato otimizado das divisões manuais, e serviu como principal benchmarking para o desenvolver da eficiência do projeto. Ao longo do tempo, após todos os ajustes, a resolução por posições variantes se provou ainda mais eficiente na grande maioria dos casos testados e portanto foi escolhida como o método definitivo.

A execução do programa ocorre a partir de dois conjuntos de endereços: um para clientes e outro para funcionários. Os endereços dos funcionários são pré-processados, enquanto os dos clientes são passados por uma ferramenta chamada de geolocalizador, que encontra para cada endereço a sua coordenada no mapa. A ferramenta se torna incapaz de encontrar um endereço caso esteja mal formatado, e portanto este é excluído da execução. Muitos erros de escrita ocorrem no processo atual do setor, e como consequência disso, 8,5% dos endereços cadastrados no banco de dados não foram localizados. Portanto, a quilometragem diária real utilizada para comparativo da eficiência foi reduzida na mesma proporção, caso contrário a simulação da operação automática envolveria menos endereços que os considerados pela média realista de quilometragem, o que tornaria os resultados da pesquisa enviesados com diferenças maiores que as reais entre um modelo e outro.

A melhoria de processo que a versão final deste software logístico propõe ocorre em 8 etapas principais. O método de operação manual executado pela cognição humana busca proximidade às tarefas listadas abaixo, mesmo que de forma menos consciente, apesar de não ter a precisão que a máquina é capaz de alcançar através do rigor processual, cálculos inequívocos e correlação de dados. São elas:

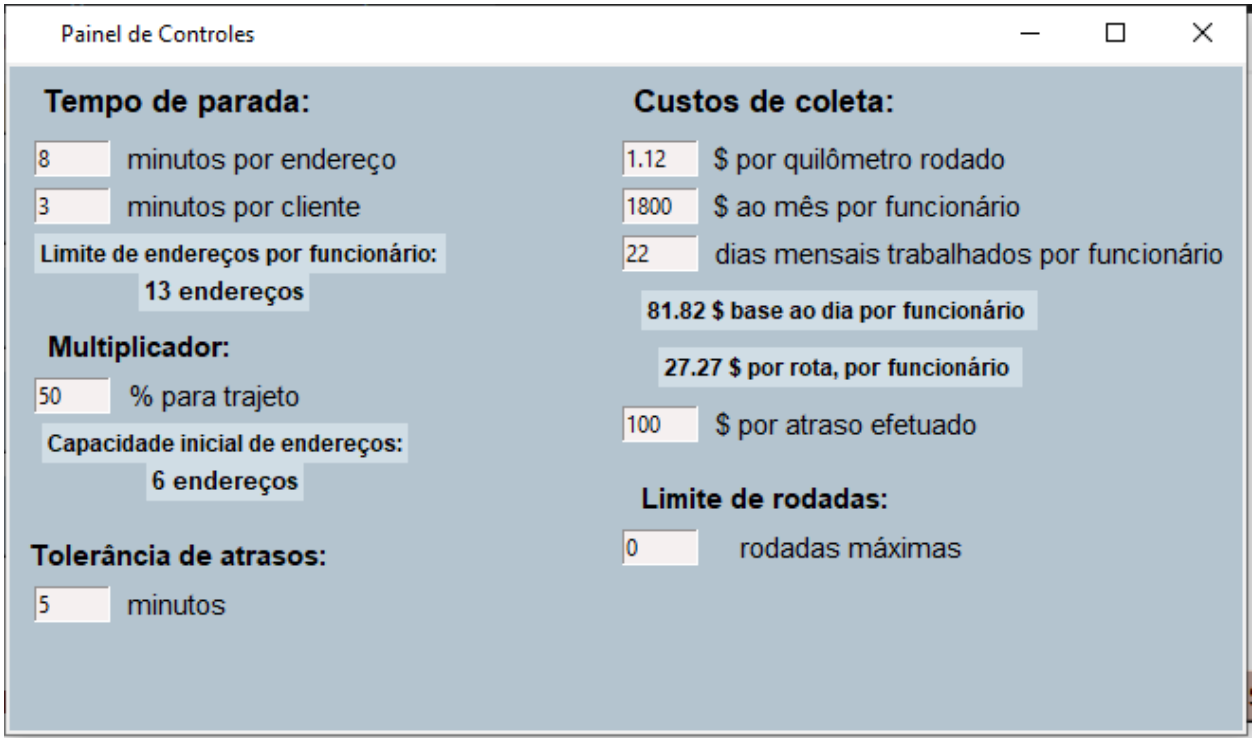
1. Estabelecer os centróides para os quais as proximidades dos clientes serão testadas (neste caso, cada centróide é o ponto inicial de um funcionário de veículos disponível).
2. Organizar a lista de clientes para a divisão, ordenando de forma decrescente desde o cliente com maior média entre as distâncias aos centróides.

3. Atribuir cada cliente ao funcionário mais próximo, cuja duração do trajeto seja menor que o limite da janela de tempo.
4. Otimizar a rota do funcionário a cada vez que um novo cliente lhe é atribuído, e mensurar o tempo total de trajeto (esta é a resolução do problema PCV mencionado anteriormente). Quando um funcionário excede o limite de tempo, torna-se indisponível para adquirir novos endereços.
5. Após todos os clientes distribuídos, mover cada centróide para a posição média entre as localizações dos clientes a este atribuídos.
6. Verificar quais funcionários têm tempos disponíveis maiores que um terço do limite da janela de tempo. Para cada um positivo nesta restrição, será testada a divisão de todos seus endereços para os demais funcionários mais próximos. A redistribuição é confirmada se: a) todos os demais funcionários permanecerem abaixo do limite de tempo ao obter os clientes redistribuídos, b) o custo total entre os trajetos for mais econômico após a redistribuição.
7. Repetir o processo acima, a partir da etapa 3.

   O processo termina após exceder um número máximo de rodadas ou quando as posições dos centróides não variam entre uma rodada e outra.

Algumas funcionalidades além do roteamento de veículos foram envolvidas no programa para auxiliar a utilização pelos funcionários. Pela interface do Resolvedor PRV, é possível acessar algumas configurações para personalizar a resolução:

Figura 10 – Interface de controle do software de resolução do PRV.



Fonte: Autoria própria.

De forma breve, as variáveis contidas no Painel de Controles funcionam das seguintes formas:

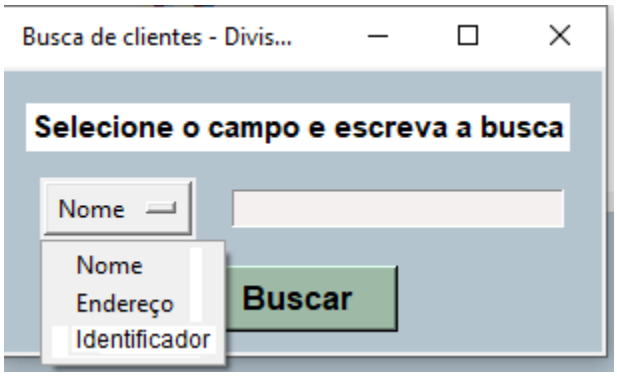
- No tempo de parada, configura-se o intervalo por localidade e o intervalo por cliente (pois em uma localidade pode haver mais do que um cliente). No caso acima, um endereço com um cliente levaria 11 minutos, enquanto um prédio com 2 clientes levaria 14 minutos.
- O multiplicador, o limite de endereços e a capacidade de endereços, como foi explicado anteriormente, não é mais usado. Mas a função está disponível e pode ser alterada para uma capacidade veicular, por exemplo.
- O campo para tolerância de atrasos permite que os tempos de trajeto previstos possam ser extrapolados na quantidade de minutos inserida.
- Nos custos de coleta, o campo para valor da quilometragem serve para verificar o preço do trajeto em termos de recurso gasto; pode ser configurado de acordo com o preço do quilômetro equivalente ao da gasolina ou com um valor acordado pela empresa. Os campos de valor por funcionário são utilizados para

que o algoritmo escolha a utilização ou não de um funcionário envolvido na operação.

- O valor por atraso efetuado pode ser utilizado para “penalizar” o algoritmo com um valor arbitrário quando houver o atraso de algum funcionário na operação, a fim de desestimular que isso ocorra nas simulações.
- O limite de rodadas pode ser utilizado inserindo um número diferente de 0, objetivando que a execução do roteamento termine assim que a rodada inserida for atingida.

Pode-se pesquisar por alguma linha de texto no banco de dados nos campos Nome, Endereço ou número identificador de um cliente. Só serão mostrados os resultados que contiverem o texto buscado na coluna selecionada:

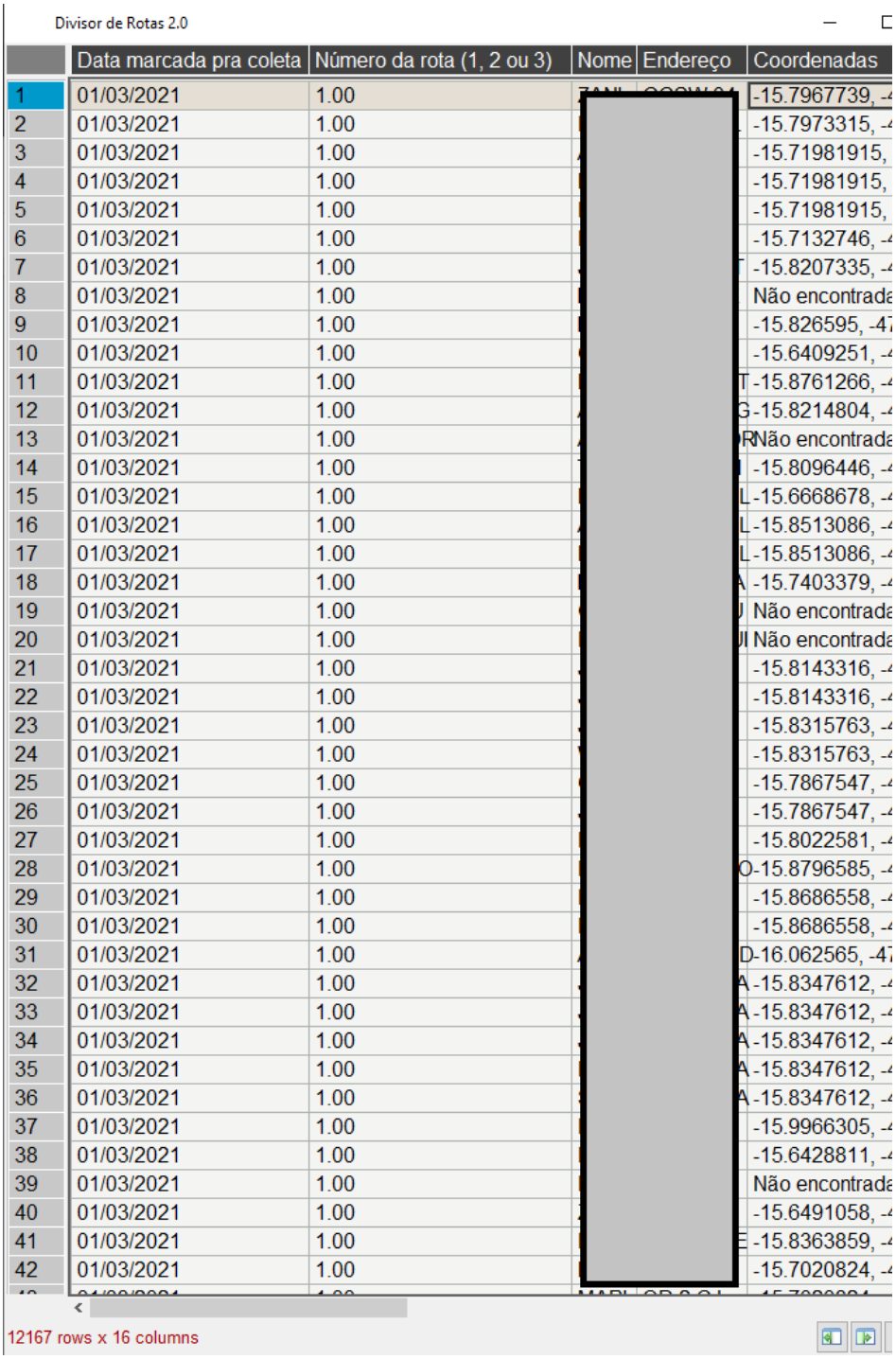
Figura 11 – Interface de busca do software de resolução do PRV.



Fonte: Autoria própria.

Também é possível acessar diretamente pelo software o banco de dados completo com os clientes cadastrados, apesar de o banco estar originalmente em uma planilha no Google Drive:

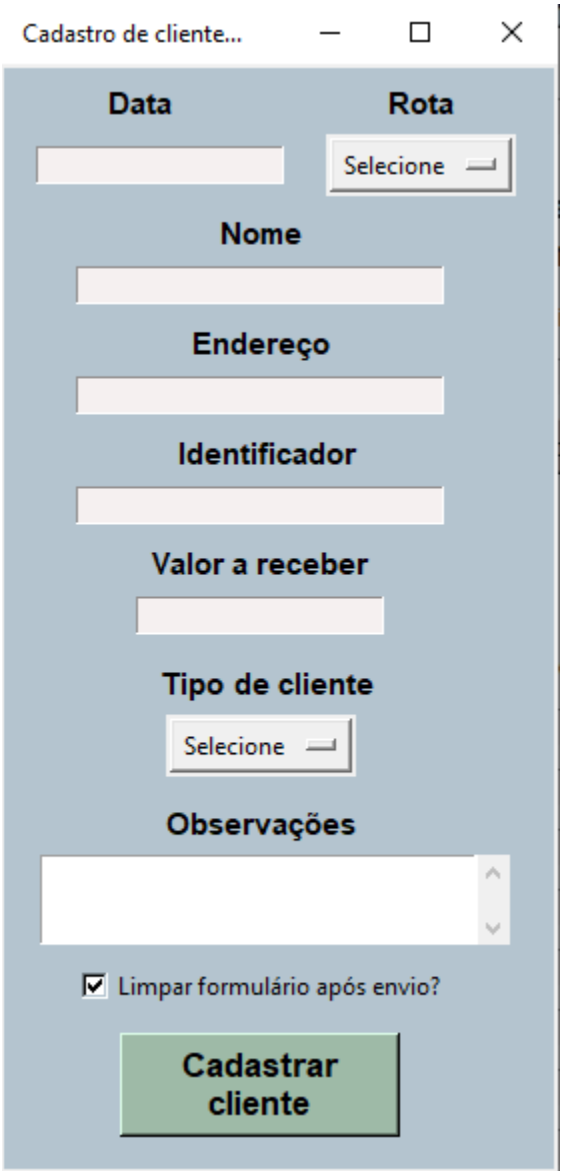
Figura 12 – Base de clientes do software de resolução do PRV. Observação: inclui tarja para preservar identidades.



Fonte: Autoria própria.

O cadastro de clientes no banco de dados pode ser feito diretamente pela planilha na nuvem (modificação direta ao banco de dados) ou então pela janela de cadastros no software:

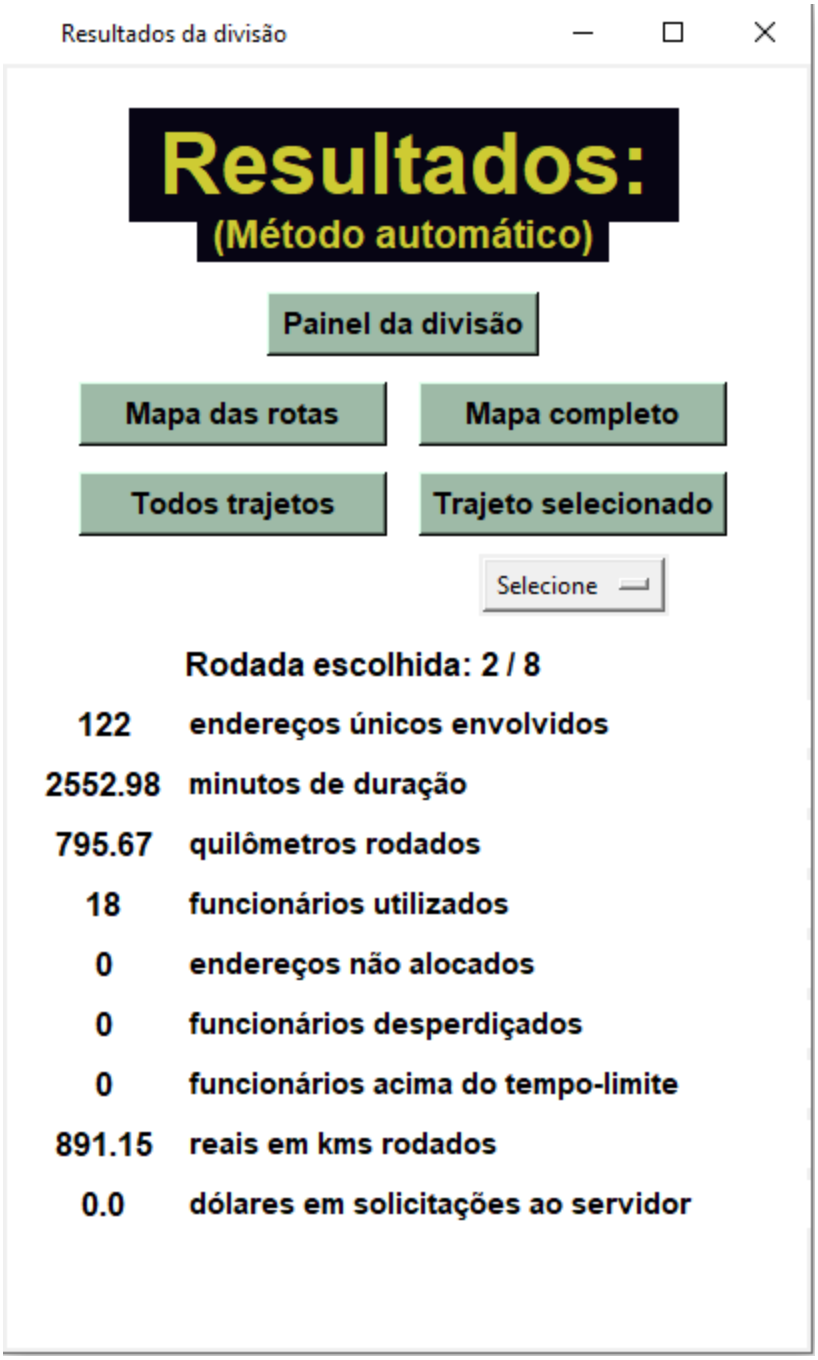
Figura 13 – Cadastro de clientes no software de resolução do PRV.



Fonte: Autoria própria.

Quando terminada a execução da construção de uma rota, surge a seguinte janela com as informações mais importantes sobre o desempenho da execução:

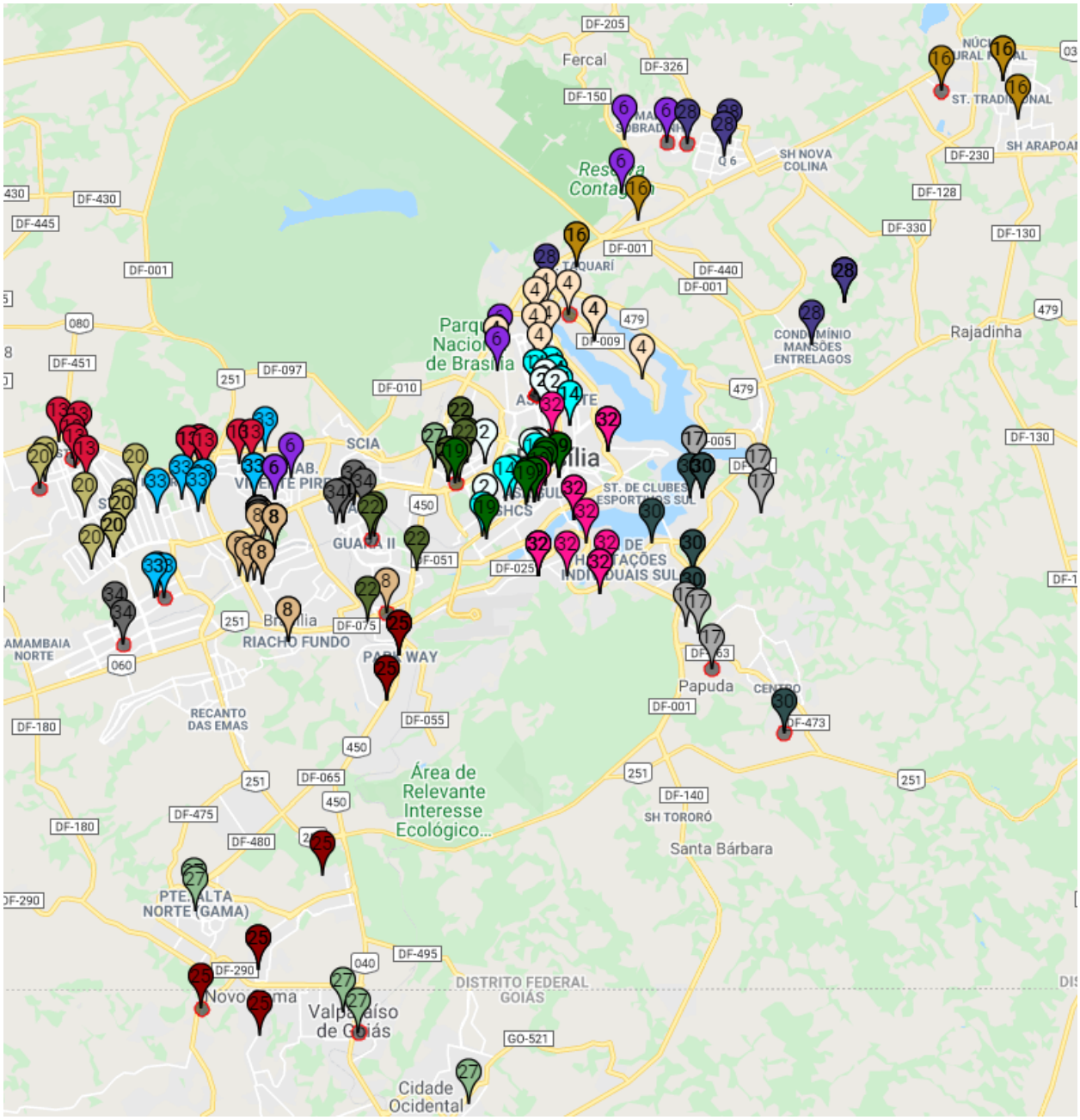
Figura 14 – Janela de Resultados do software de resolução do PRV.



Fonte: Autoria própria.

Através da janela de resultados, o botão “Mapa das rotas” mostra como ficou a divisão dos locais entre os funcionários disponíveis, cada um representado por uma cor distinta:

Figura 15 – Mapa dos conjuntos de endereços no software de resolução do PRV.

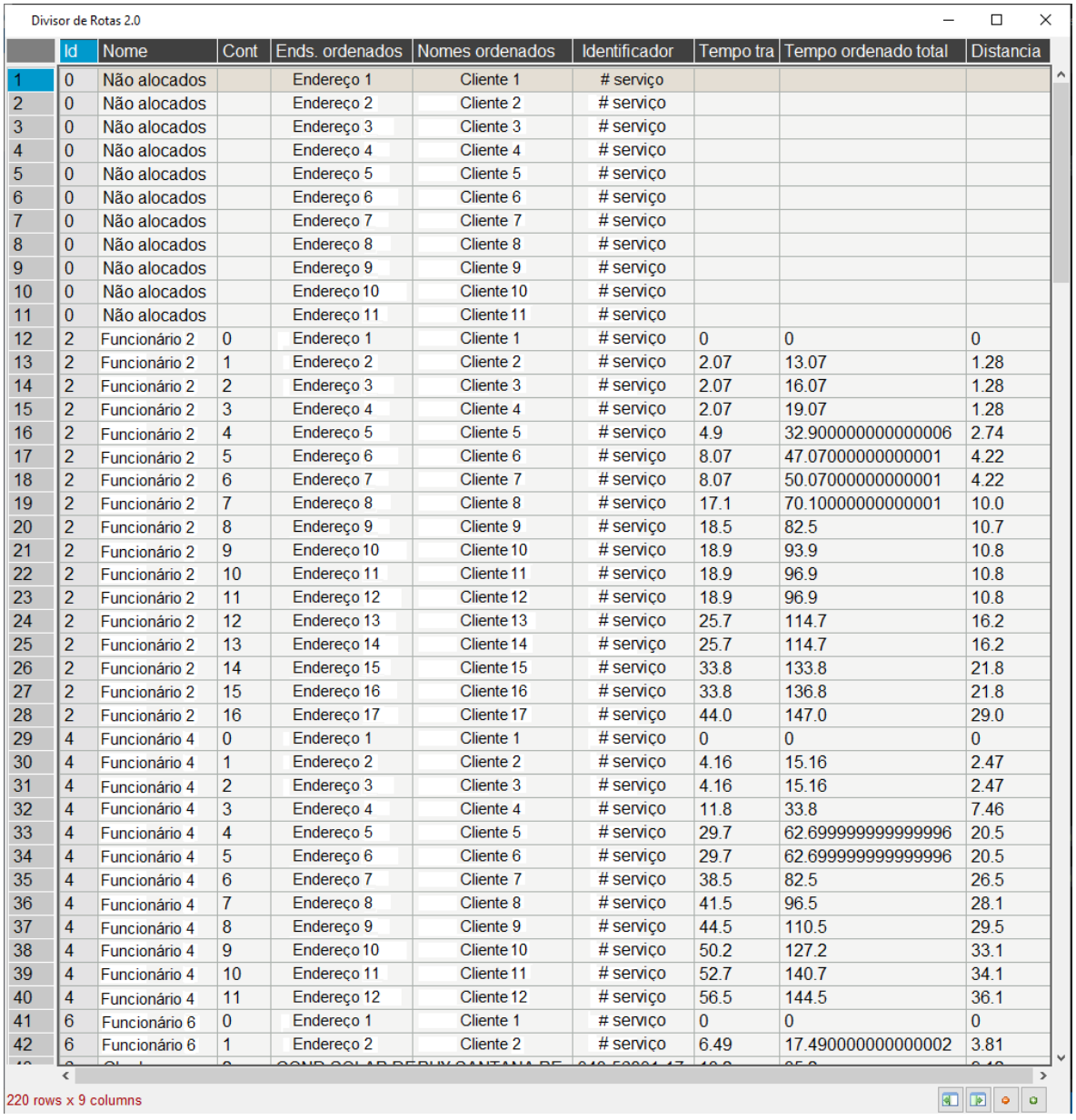


Fonte: Autoria própria.

O botão “Painel da divisão” mostra o painel final com as informações mais importantes sobre os trajetos da operação, como os endereços alocados a cada funcionário na ordem que devem ser percorridos, e suas distâncias e durações cumulativas conforme o andamento da atividade. As células que mostram “Não

alocados” correspondem àqueles endereços que o geolocalizador não foi capaz de encontrar a respectiva coordenada no mapa (devido à má formatação do endereço), e portanto podem ser alocados manualmente pelos operadores ou corrigidos antes de executar novamente:

Figura 16 – Painel resultante da execução, contendo as informações chave da operação.



	Id	Nome	Cont	Ends. ordenados	Nomes ordenados	Identificador	Tempo tra	Tempo ordenado total	Distancia
1	0	Não alocados		Endereço 1	Cliente 1	# serviço			
2	0	Não alocados		Endereço 2	Cliente 2	# serviço			
3	0	Não alocados		Endereço 3	Cliente 3	# serviço			
4	0	Não alocados		Endereço 4	Cliente 4	# serviço			
5	0	Não alocados		Endereço 5	Cliente 5	# serviço			
6	0	Não alocados		Endereço 6	Cliente 6	# serviço			
7	0	Não alocados		Endereço 7	Cliente 7	# serviço			
8	0	Não alocados		Endereço 8	Cliente 8	# serviço			
9	0	Não alocados		Endereço 9	Cliente 9	# serviço			
10	0	Não alocados		Endereço 10	Cliente 10	# serviço			
11	0	Não alocados		Endereço 11	Cliente 11	# serviço			
12	2	Funcionário 2	0	Endereço 1	Cliente 1	# serviço	0	0	0
13	2	Funcionário 2	1	Endereço 2	Cliente 2	# serviço	2.07	13.07	1.28
14	2	Funcionário 2	2	Endereço 3	Cliente 3	# serviço	2.07	16.07	1.28
15	2	Funcionário 2	3	Endereço 4	Cliente 4	# serviço	2.07	19.07	1.28
16	2	Funcionário 2	4	Endereço 5	Cliente 5	# serviço	4.9	32.900000000000006	2.74
17	2	Funcionário 2	5	Endereço 6	Cliente 6	# serviço	8.07	47.07000000000001	4.22
18	2	Funcionário 2	6	Endereço 7	Cliente 7	# serviço	8.07	50.07000000000001	4.22
19	2	Funcionário 2	7	Endereço 8	Cliente 8	# serviço	17.1	70.10000000000001	10.0
20	2	Funcionário 2	8	Endereço 9	Cliente 9	# serviço	18.5	82.5	10.7
21	2	Funcionário 2	9	Endereço 10	Cliente 10	# serviço	18.9	93.9	10.8
22	2	Funcionário 2	10	Endereço 11	Cliente 11	# serviço	18.9	96.9	10.8
23	2	Funcionário 2	11	Endereço 12	Cliente 12	# serviço	18.9	96.9	10.8
24	2	Funcionário 2	12	Endereço 13	Cliente 13	# serviço	25.7	114.7	16.2
25	2	Funcionário 2	13	Endereço 14	Cliente 14	# serviço	25.7	114.7	16.2
26	2	Funcionário 2	14	Endereço 15	Cliente 15	# serviço	33.8	133.8	21.8
27	2	Funcionário 2	15	Endereço 16	Cliente 16	# serviço	33.8	136.8	21.8
28	2	Funcionário 2	16	Endereço 17	Cliente 17	# serviço	44.0	147.0	29.0
29	4	Funcionário 4	0	Endereço 1	Cliente 1	# serviço	0	0	0
30	4	Funcionário 4	1	Endereço 2	Cliente 2	# serviço	4.16	15.16	2.47
31	4	Funcionário 4	2	Endereço 3	Cliente 3	# serviço	4.16	15.16	2.47
32	4	Funcionário 4	3	Endereço 4	Cliente 4	# serviço	11.8	33.8	7.46
33	4	Funcionário 4	4	Endereço 5	Cliente 5	# serviço	29.7	62.699999999999996	20.5
34	4	Funcionário 4	5	Endereço 6	Cliente 6	# serviço	29.7	62.699999999999996	20.5
35	4	Funcionário 4	6	Endereço 7	Cliente 7	# serviço	38.5	82.5	26.5
36	4	Funcionário 4	7	Endereço 8	Cliente 8	# serviço	41.5	96.5	28.1
37	4	Funcionário 4	8	Endereço 9	Cliente 9	# serviço	44.5	110.5	29.5
38	4	Funcionário 4	9	Endereço 10	Cliente 10	# serviço	50.2	127.2	33.1
39	4	Funcionário 4	10	Endereço 11	Cliente 11	# serviço	52.7	140.7	34.1
40	4	Funcionário 4	11	Endereço 12	Cliente 12	# serviço	56.5	144.5	36.1
41	6	Funcionário 6	0	Endereço 1	Cliente 1	# serviço	0	0	0
42	6	Funcionário 6	1	Endereço 2	Cliente 2	# serviço	6.49	17.490000000000002	3.81

Fonte: Autoria própria.

As informações contidas na figura acima foram substituídas para a preservação de dados pessoais. A coluna “Nome” contém o funcionário responsável pelo cliente mencionado, a coluna “Cont” numera a contagem de endereços de um funcionário, em “Ends. ordenados” estão contidos os endereços que cada funcionário deve percorrer, sendo que o primeiro endereço de cada funcionário (numerado 0) é o seu local de partida, e o último é o local de finalização. A coluna “Nomes ordenados” contém o nome do cliente que será atendido naquele endereço, assim como a coluna “Identificador” contém um número único para identificar o atendimento. A coluna “Tempo trajeto” apresenta em minutos o acumulado de tempo de duração da atividade, considerando apenas o deslocamento (sem considerar os tempos de parada). Em “Tempo ordenado total” são mostrados os tempos cumulativos de trajeto somados aos tempos de parada. E por fim, a coluna “Distância” mostra em quilômetros o deslocamento do veículo até o momento final da operação.

Para mensuração dos resultados, foi construída uma planilha de desempenhos contendo as informações chave de cada execução.

Figura 17 – Matriz de resultados, delimitada por método, dia e jornada.

	Metodo	Data	Rota	Distancia total	Duracao total	Atrasado	Custo do trajeto	Qtd. rodadas	Total rodadas
0	auto	01/03/2021	1	750.0318	2502.935000	0	840.035616	3.0	8.0
20	manual	01/03/2021	1	1119.6540	2974.040000	5	1254.012480	0.0	0.0
45	auto	01/03/2021	2	534.9454	1374.058333	0	599.138848	2.0	6.0
64	manual	01/03/2021	2	1105.5261	1989.195000	9	1238.189232	0.0	0.0
85	auto	01/03/2021	3	783.4424	2172.486667	1	877.455488	29.0	32.0
...	...	...	...	...	...	...	...	...	...
3439	manual	31/03/2021	1	1292.2997	3425.516667	6	1447.375664	0.0	0.0
3467	auto	31/03/2021	2	427.6449	1181.375000	0	478.962288	3.0	5.0
3482	manual	31/03/2021	2	918.0326	1722.656667	7	1028.196512	0.0	0.0
3502	auto	31/03/2021	3	958.2983	2917.473333	0	1073.294096	5.0	32.0
3528	manual	31/03/2021	3	1452.2271	3482.661667	13	1626.494352	0.0	0.0

170 rows × 9 columns

Fonte: Autoria própria

Antes da verificação dos percentuais de ganho, uma avaliação foi feita para cada dia e rota planilhados, para checar se em alguma ocasião o método automático teve um desempenho inferior. Foi encontrado que em todas as resoluções, o automático desempenhou melhor que o método manual otimizado, caracterizado na figura apenas por “manual”.

Os desempenhos gerais de tempo pelo método automático foram verificados somando a coluna ‘Duração total’ para toda ocorrência em que o valor da coluna “Método” fosse ‘auto’. O mesmo foi feito para as ocorrências em que a coluna “Método” fosse ‘manual’. Considerando que os valores ‘auto’ são sempre menores, a subtração da soma ‘manual’ pela soma ‘auto’ resulta no valor do tempo ganho. O ganho percentual é obtido pela conta:

Ganho percentual = 1 - (tempo ganho / soma ‘manual’)

Estes são os resultados para o indicador de Tempo Ganho; o mesmo foi feito para o indicador de Distância Economizada, repetindo o processo acima utilizando os valores da coluna “Distância total” ao invés da coluna de duração.

6. Resultados

De acordo com o que se propõe pela gestão de processos, incluindo a remodelagem, o desenvolvimento organizacional tem como base objetiva o foco em alcançar melhorias de desempenho nos processos (Laurindo e Rotondaro, 2006). Por conta disso, a projeção do aplicativo teve um enfoque na evolução do algoritmo para obtenção de resultados positivos, no mínimo na maioria das execuções, em todos os indicadores levantados.

Os resultados obtidos desde a operação manual, como a simulação da operação manual, para a operação automática são muito convincentes. Sobre o desempenho do produto aqui relatado, não há conclusão quanto à qualidade da solução em relação a outros softwares semelhantes pois não houve comparação, mas existe a comprovação, através da verificação empírica por simulação, de que o produto supera as expectativas iniciais e garante, de forma gratuita, uma qualidade da operação no mínimo mais barata e mais eficiente que a atual, reduzindo os atrasos, aumentando o controle sobre a operação, economizando custos e aumentando a capacidade operacional.

Uma ampla verificação de desempenho do software foi necessária para decidir quanto à sua utilização na operação corporativa. Para tal foi efetuada uma pesquisa operacional retroativa visando comparar o formato original com o formato automático do processo de logística de rotas, utilizando dados históricos da operação. Os dados históricos são compostos por registros dos blocos de endereços visitados por cada funcionário, pela quantidade de funcionários utilizados e pela média da quilometragem diária total percorrida, dados que compõem o modelo chamado de “realista”. A mensuração dos desempenhos durou aproximadamente 20 horas de processamento, e foi feita a partir de simulações das rotas de todos os dias de março. Devido ao histórico ter registros apenas de quais endereços os funcionários visitaram, mas não da ordem entre eles e nem quanto tempo cada um levou para completá-los, foi incluído na pesquisa um terceiro formato chamado de “divisões manuais otimizadas”. Consiste na consideração de um cenário otimista da operação que ocorreu, utilizando os blocos de endereços divididos manualmente mas pressupondo a melhor ordem entre eles para a mensuração das distâncias e durações. O modelo automático, por sua vez, refaz toda a

divisão dos endereços buscando minimizar a distância percorrida, a duração tomada e a quantidade de funcionários utilizados na operação. Temos portanto 3 modelos para comparação:

a) O formato realista (ou formato manual), com dados reais das distâncias diárias que a operação “custou” e a quantidade de funcionários envolvidos em cada dia.
b) O formato realista otimista (também chamado de formato manual otimizado), com dados simulados das distâncias e durações caso os funcionários envolvidos tivessem utilizado apenas os caminhos mais curtos.
c) E o formato automático, com dados de distâncias, durações e funcionários utilizados a partir dos caminhos mais curtos entre os blocos de endereços reconstruídos.

6.1 Modelo realista x Modelo realista otimizado

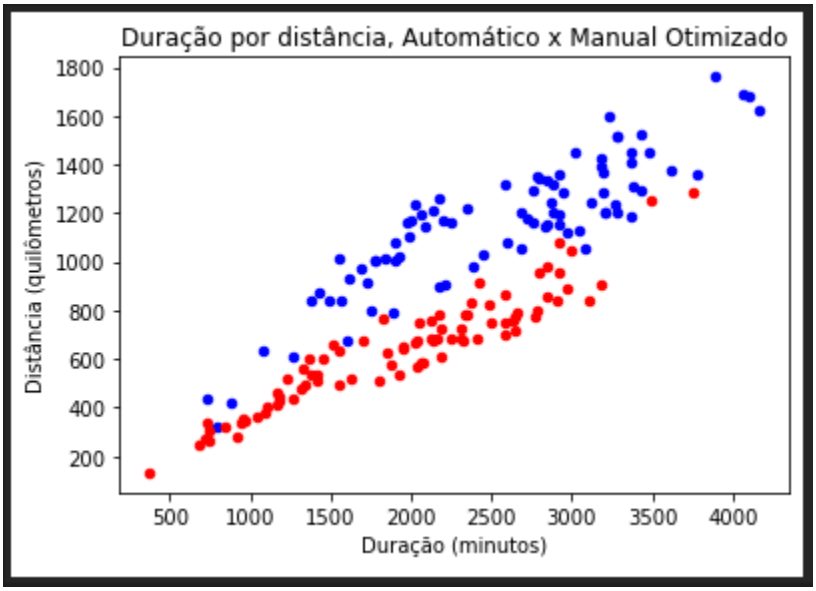
A operação original utilizou uma média diária de 21.6 veículos e funcionários, e 4400 quilômetros percorridos. Com o reajuste de 8,5% devido ao problema das coordenadas não localizadas explicado anteriormente no tópico 4) Desenvolvimento, a média realista considerada passa a ser 4026 km por dia. A operação manual otimizada, por sua vez, apontou uma média de 3173 km utilizando os mesmos funcionários, que poderia ter sido alcançada caso todos os percursos houvessem sido feitos nos trajetos mais curtos, o que demonstra um ganho de 21,2% apenas por otimizar as ordens entre os endereços (sem precisar reformular as atribuições).

Como não temos dados da operação realista além da quilometragem e funcionários utilizados, o desempenho do formato automático também é comparado em relação ao formato manual otimizado, pois pretendemos observar inclusive os ganhos em capacidade produtiva (por economia de tempo e funcionários), e em quantidade de atrasos (funcionários que extrapolam o horário de tempo em algum turno). A comparação entre o modelo automático e o modelo realista otimizado também permite a desconsideração de adversidades processuais (como por exemplo pneu furado ou

chuvas fortes), visto que as diferenças entre ambos utilizam as mesmas condições e métricas operacionais.

6.2 Resultados do Modelo automático

Gráfico 3 – Resultados das simulações do mês, modelo automático (vermelho) versus modelo manual otimizado (azul)



Fonte: Autoria própria.

O gráfico acima compila os resultados gerais das execuções das jornadas de trabalho entre os dias 1 e 31 do mês testado, em termos de duração e distância. Por se tratar de um problema de minimização de recursos gastos, temos a meta do tipo “menor, melhor”: quanto mais próximo do ponto 0 (vértice inferior esquerdo), mais eficiente foi a referida resolução. Todas as jornadas efetuadas têm duas marcações, uma vermelha e uma azul, referentes aos dois métodos que foram utilizados. Nota-se que o conjunto de resoluções pelo método automático tem uma dispersão menor, sendo mais linear, e tem mínimos e máximos menores que o método manual otimizado. De forma geral, se vê que os resultados em vermelho foram menos custosos em tempo gasto e trajeto percorrido.

Em quilometragem rodada, o formato automático da operação desempenhou uma média de 1761 quilômetros por dia, um ganho de 56,2% em relação ao formato realista e 44,5% em relação ao formato realista otimizado.

Em economia de recursos humanos, o automático apresentou média de 16,6 funcionários diários, uma diferença de 5 funcionários utilizados, ganho equivalente a 23,1% no consumo de RH.

Em tempo de trajeto, o modelo manual otimizado obteve uma média de 7013 minutos diários de operação versus 5332 minutos para o modelo automático, ganho de 23,9% em disponibilidade de tempo.

Em pontualidade dos funcionários, considerando como atrasado o funcionário cujo tempo de trajeto mensurado excede o limite de tempo proposto pela escala de trabalho com tolerância de 3 minutos, o modelo automático apresenta uma melhoria de 98,2% em relação ao modelo manual otimizado. Durante todas as escalas do mês mensurado, o modelo automático excedeu o limite de tempo 10 vezes, contra 549 vezes para o modelo manual otimizado. Observa-se que se a operação realista tivesse dados de atrasos, seu número seria certamente maior, tendo em vista que as durações de trajeto são invariavelmente maiores no modelo realista relativamente ao modelo realista otimizado.

Resta considerar que as melhorias dispostas pelo modelo automático são pelo menos as demonstradas pelas comparativas versus o manual otimizado, permitindo uma interpretação de que a aplicação do novo modelo automatizado de operação na empresa retornará resultados maiores que os observados, considerando um cenário que não haja interferências externas no procedimento ideal.

Em resumo, os ganhos apresentados na simulação pela automação do software comparado à operação original são:

- 56,2% de economia em quilometragem e gasolina;
- 23,1% de economia em recursos humanos da linha de frente;
- 23,9% de economia em tempo gasto na operação;
- 98,2% de redução de atrasos que ultrapassaram o limite de tempo.

Para verificar os ganhos em termos financeiros, consideremos um ticket médio de R$ 305,00 e uma política institucional de ressarcimento de R$1,12 por quilômetro rodado, sendo os funcionários responsáveis pela gasolina dos veículos. Com isso, observamos as diferenças: a operação real teve um custo por trajeto percorrido de R$139.782,00 aos cofres da empresa, enquanto no formato automático de atribuição teria custado apenas R$61.159,00, um ganho de R$78.623,00 mensais. A economia de custos não considera os funcionários poupados e seus salários, pois entende-se que estes não serão dispensados, mas sim alocados para atender mais clientes incrementando a capacidade produtiva com mais oferta disponível.

Em ganho de disponibilidade convertido para faturamento, considerando um cenário onde há demanda para toda a oferta disponibilizada, temos os incrementos:

a) dos 23,9% de economia em tempo gasto;
b) dos tempos dos funcionários economizados ao longo do mês.

Esses tempos são quantificados em minutos e multiplicados pelo ticket médio vezes a quantidade de clientes atendidos por minuto. Portanto temos:

- Tempo ganho: 108164.135 minutos
    - Tempo ganho por otimização dos trajetos: 52124.135 minutos
    - Tempo ganho por economia de funcionários: 56040 minutos
- Clientes atendidos por minuto de operação: 0.04972 atendimentos/minuto
- Ticket médio: R$305,00

*Acréscimo previsto ao faturamento com a expansão do atendimento no tempo incremental: 108164.135 * 0.04972 * 305 = <u>R$1.640.265,84.</u>*

Ganhos financeiros mensais por referência de março/2021:

- R$1.640.265,84 a mais de faturamento via aumento da capacidade produtiva, considerando demanda disponível para toda oferta disponibilizável;
- R$78.623,00 a mais de lucro via economia de custos.

Cabe ressaltar que a otimização do desempenho em um processo como este pende sobre a gratificação correta do funcionário de linha de frente. Uma remuneração extra por cliente atendido, por exemplo, tende a maximizar a produção dos atendimentos, enquanto uma remuneração exclusivamente por quilômetros percorridos tende a aumentar os riscos operacionais devido às diferenças dos objetivos de maximização de retorno entre o funcionário e a empresa, conforme aponta o estudo sociológico conceituado como *homo economicus* ou Homem Econômico (Taylor, 1911 apud Wagner-Tsukamoto, 2018), abordado na Teoria da Administração Científica de Taylor (Taylorismo), e presente na Teoria dos Jogos e Teoria das Escolhas Racionais. Com isso, conclui-se que um aumento na alocação de custos em formato de benefícios aos funcionários de frente tende, como consequência, a elevar o faturamento da operação devido ao aumento do desempenho da divisão.

Indo além dos ganhos financeiros, o software incrementa o controle da operação com um painel de acompanhamento das previsões diárias, introduzindo à operação uma gestão baseada em métricas absolutas e permitindo pareceres aos clientes que tenham mais pressa: ao que a empresa recebe contato de um cliente marcado, o atendente de retaguarda pode verificar no software o tempo faltante previsto para que o funcionário delegado chegue ao local deste cliente, resultando em um incremento ao consumidor pela disposição de mais informações sobre a operação, e na qualidade de trabalho do operador através de mais facilidade e domínio sobre a atividade. A gestão por métricas também garante ao coordenador uma capacidade maior de controlar o desempenho do seu setor, tanto de forma geral para verificar as variações no decorrer do tempo, como de forma individualizada em eficiência de atendimento e quilometragem utilizada por funcionário.

Outra vantagem operacional garantida pela introdução do software ao procedimento é o ganho de qualidade no exercício, devido à redução de 98% nos atrasos. Apesar de ser um ganho imensurável, é plausível esperar uma melhoria na satisfação geral dos clientes com a marca, baseada na confiança ao serviço prestado, e portanto também um aumento da demanda com a expansão do público disposto a consumir e da fidelização de consumidores.

Como mencionado anteriormente, o software também garante uma grande redução na carga de trabalho de aproximadamente 12 funcionários de retaguarda responsáveis por formular os trajetos na operação manual de construção das rotas. No entanto, esses ganhos não foram incluídos nos ganhos financeiros relatados pois não houve a mensuração quanto ao dimensionamento da equipe para a operação *to-be* (pós-implementação do programa). Isso se deve principalmente ao fato de que, apesar de o roteamento dos trajetos ser a atividade da área de maior foco, outras subatividades também estão envolvidas no processo e portanto o dimensionamento será mais eficaz e realista com a mensuração sendo feita efetivamente após a inclusão do software ao método. De forma simplória, estima-se uma redução de ao menos 50% da carga de trabalho da área.

Por fim, há também o ganho pela redução de emissão veicular ao meio ambiente (não mensurado), que acompanha os 70.200 quilômetros de trajeto potencialmente economizados por mês.

6.3 Diferencial do sistema

O desenvolvimento do próprio algoritmo para a resolução da operação institucional exerce um impacto positivo no processo por apresentar alguns diferenciais quanto a outros *softwares* verificados. Em relação a ferramentas de código aberto para a resolução do PRV, este software permitiu:

a) Fazer o roteamento de um ou mais veículos,
b) Estabelecer limites de tempo,
c) Definir capacidades do veículo,
d) Definir custos da operação,
e) Buscar a melhor rota baseando-se no custo de trajeto,
f) Escolher entre minimizar a duração ou a distância.

Nenhuma das ferramentas *open source* analisadas permitiu o manejo de todas essas variáveis, portanto a utilização do algoritmo particular se provou mais útil à operação por permitir uma atividade mais customizada e controlável.

6.4 Limitações do projeto

Este programa se inaugurou com a intenção inicial de automatizar um processo corporativo para permitir uma operação mais simples e fácil aos funcionários de retaguarda. Os resultados finais foram satisfatórios pois o software alcançou além do que era pretendido. Posto isso, as práticas aqui expostas podem não competir com resoluções construídas por profissionais do ramo de logística ou programadores experientes com prática em rotas veiculares, pois essa comparativa não foi testada no período hábil deste projeto. A eficiência testada foi em relação ao formato original (manual) da operação e ao formato otimizado da operação original, para a formulação de um “cenário perfeito” a ser comparado com o formato automático.

Uma limitação da execução do aplicativo é a desconsideração do trânsito em tempo real para verificação dos melhores trajetos e mensuração de tempos e distâncias. Os dados utilizados são históricos, portanto uma operação que passe a contemplar com precisão o trânsito da cidade no período de utilização exige a captação desses dados. Para o presente projeto, esta etapa poderá ser iniciada após a implementação do aplicativo à operação, captando dados por meio de GPS dos veículos em atividade e da comparação dos desempenhos previstos em relação aos efetuados.

Outra particularidade do software em comparação ao formato da operação original é a limitação em entregar atendimentos de forma igualitária aos funcionários. Como a divisão é feita de forma automática, a alocação de um cliente envolve a proximidade dos funcionários e os tempos disponíveis de cada um, de forma que um funcionário possa receber, por exemplo, 9 locais para visitar e outro apenas 1. Um desenvolvimento do sistema seria possível no sentido de equilibrar a alocação de atendimentos contemplados, mas isso não foi percebido como benéfico na perspectiva de otimização da operação pois desviaria o foco em economizar caminho percorrido e tempo de trabalho, o que resultaria em um ganho menos significante da capacidade operacional.

A escassez de informações quanto à presença de cada funcionário ao longo dos dias também é uma limitação do projeto, pois as simulações trataram todos os funcionários como disponíveis. Isto é uma limitação pois os resultados se baseiam na comparação entre os formatos de operação automática e de operação original, enquanto não se sabe se os funcionários escolhidos pela automática estavam disponíveis durante a ocorrência da original. Esses casos são em geral raros, pois os funcionários costumam estar disponíveis, mas caso tenha ocorrido, sabe-se que a máquina optou pela provável melhor escolha enquanto a operação originalmente ocorrida não teve a mesma oportunidade.

7. Sugestões e possíveis melhorias

O programa desenvolvido até aqui agrega funções que podem ser destrinchadas de acordo com cinco níveis de produto que garantem o seu valor, propostos por Philip Kotler (CGMA, 2021). O primeiro nível é o benefício fundamental do produto, que satisfaz a necessidade absoluta (neste caso, o roteamento de veículos em si). O segundo nível é o produto genérico, que envolve os atributos necessários para um funcionamento correto do programa de acordo com os objetivos (variáveis de intervalo de tempo, por exemplo). O terceiro nível se refere ao produto esperado, que satisfaz adequadamente as expectativas de quem o utiliza. No caso do software, é um roteador de trajetos que seja mais eficiente que a operação original. O quarto e penúltimo nível se chama produto aumentado, com especificidades que o diferenciam de outros produtos semelhantes. Neste caso, as facilidades de cadastro e busca de clientes, visualização de banco de dados, seleção da prioridade para otimização (distância, duração ou custos) ou visualização da operação no mapa podem ser percebidos como parte de um produto aumentado. E por fim, no quinto nível, o produto potencial se refere às capacidades que a mercadoria pode alcançar mas ainda não as abrange. São as funções do quinto nível da escala de produtos de Kotler que serão abordados abaixo.

Sugestões para futuros estudos envolvem a verificação e descrição em português dos métodos e heurísticas mais eficientes no roteamento de múltiplos veículos. Identificamos possíveis melhorias que podem ser incluídas no software apresentado de forma a agregar a operação, que serão discorridas à frente. A versão do software vigente, neste momento de escrita, não envolve as funcionalidades descritas a seguir devido à limitação de tempo hábil para implementar tudo que é de interesse para a formulação de um processo ainda melhor. O ajuste da operação correta do algoritmo foi o principal foco durante este projeto, e a partir daqui as funcionalidades extras a seguir poderão ser implementadas.

7.1 Clientes prioritários

Uma das principais utilidades para o bom andamento da operação é a possibilidade de definir endereços como prioridade. Com isso, será possível definir que um cliente seja visitado antes que todos os outros ou no máximo até tal horário, de forma a “ignorar” a otimização num período inicial. Isso será feito com a introdução da variável “prioridade” a cada um dos funcionários da linha de frente, que será uma lista podendo conter endereços. Se não for vazia para um funcionário, o primeiro passo do algoritmo será verificar a distância e duração do ponto inicial deste funcionário até o local, tornando em seguida este local como o seu ponto inicial.

7.2 Interface de usuário avançada

Outra funcionalidade interessante é a criação de uma interface com atualização em tempo real conforme se passa o tempo da operação. Com isso, o operador de retaguarda conseguirá verificar no mapa qual é o local projetado que o funcionário deve estar, de acordo com o tempo extrapolado desde o início da atividade. Essa funcionalidade fica ainda mais precisa com a utilização de GPS nos veículos, a partir de que o operador sempre saberá as localizações exatas dos carros e terá uma previsão certeira do tempo que levará para o atendimento de cada cliente.

7.3 Evolução do formato de atendimento

Com a introdução do aplicativo à operação e alguns ajustes internos, a atividade de atendimento pode evoluir do formato atual de marcação por escalas de trabalho para um formato de horário marcado, pois agora a operação inclui métricas exatas de tempo. Dessa forma, não apenas mais clientes serão atendidos em um período mais curto, mas também poderá haver mais conforto por parte do cliente em definir exatamente em qual momento do dia ele pretende ser atendido.

Esta solução, em conjunto com uma boa interface para o operador, permitirá um processo em que o cliente escolha o atendimento em qualquer horário do seu dia de preferência ou no dia disponível mais próximo, trazendo mais conforto e agilidade ao cliente.

7.4 Encaixes para maximização de atendimentos

Uma operação que contemple o maior número de clientes atendidos tem como base um controle bem feito. Em um caso de demanda superior à capacidade de oferta, o ajuste da operação ocorreria de forma que os funcionários de linha de frente passassem praticamente todo o período da jornada de trabalho em campo. Neste caso, os operadores de retaguarda efetuariam marcações imediatas dos clientes que entrassem em contato, e o software estaria a todo momento reformulando as rotas dos veículos, que as receberiam em tempo real.

Esta solução contempla basicamente a integração do atual software a um sistema de encaixes, onde as visitas dos funcionários estariam em constante expansão durante a operação. Neste caso, os funcionários teriam consigo impressoras portáteis para geração em campo de recibos e outros documentos importantes aos clientes.

7.5 Incremento dos benefícios aos funcionários

A partir do momento em que se envolve um ganho por atendimento ao funcionário responsável pelo atendimento em domicílio, e se estabelece uma política para controle da qualidade dos atendimentos, passa a ser absoluto o interesse mútuo de funcionário-empresa em maximizar as vendas. Posto isso, é benéfico que a empresa estabeleça uma política baseada em gestão por desempenho, de forma a conceder incrementos salariais conforme o número de clientes atendidos por um funcionário.

7.6 Definição autônoma dos locais de início

Com o modelo de processo proposto pela aplicação, passa a ser possível um sistema voltado à qualidade de trabalho do funcionário baseado na escolha própria de qual local ele deseja iniciar as atividades do dia. Esta funcionalidade não estaria disponível para qualquer caso da operação, visto que às vezes é importante que um colaborador esteja disponível próximo de uma região específica; mas em determinadas ocasiões, a depender das posições dos clientes, é possível permitir que definam outros locais de início além das localizações padrão. Outro impacto gerado por esta solução seria a capacidade do funcionário em aumentar ou diminuir os clientes a si alocados,

variando seu local de início entre as regiões baseando-se na demografia do público da empresa.

8. Considerações finais

Este projeto científico demonstra que tanto na esfera nacional como ao redor do mundo, a utilização da tecnologia da programação como meio de resolução de problemas organizacionais produz desde benefícios econômicos, até na eficiência da operação, na gestão sobre os funcionários e no atendimento ao cliente, melhorando aspectos como a facilidade em executar a operação e a satisfação do cliente final. O crescimento da disponibilização de tecnologias é uma excepcional oportunidade que as empresas atuais têm em gerar diferenciais competitivos quanto ao mercado em que atuam, pois efetuar operações com tecnologias de ponta, assemelhando-se às maiores empresas internacionais do ramo, é efetivamente um dos melhores caminhos para a prosperidade financeira de qualquer organização competitiva do âmbito nacional.

A melhoria de processos, segundo Scartezini (2009), tem como grande objetivo agregar valor aos produtos e aos serviços que as organizações prestam aos seus clientes. Os ganhos previstos para a empresa-alvo, com a melhoria a partir da tecnologia da programação, são de mais de 1 milhão e meio de reais mensais no faturamento e mais de 75 mil reais de economia. Os clientes ganharão mais qualidade no atendimento, enquanto os funcionários terão mais satisfação no trabalho. Assim, constata-se que a proposta do software como uma ferramenta promotora da melhoria de processos foi bem sucedida.

Aos futuros administradores, a participação em um curso prático ou matéria voltada à computação (como Introdução à Ciência da Computação) é capaz de causar ganhos no que trata à capacidade de aplicação daquilo que é ensinado na faculdade de Administração. Esses métodos de execução computacionais excedem a teoria, e se aprofundam de forma prática naquilo que a academia instrui. Assim, enquanto o currículo de administração por vezes possa limitar a capacitação até “o que fazer”, a programação é um excelente método para compreender efetivamente “como fazer”.

Restou esclarecido, a partir dos resultados obtidos, que os objetivos deste projeto foram plenamente alcançados. A confecção automática da logística urbana provou-se útil em todos os âmbitos analisados e permitiu melhorias além das expectativas, enquanto a construção de algoritmos para resolução de problemas

computacionais simples ou complexos provou-se alinhada aos objetivos comuns de corporações, como um meio mais econômico e mais eficiente de efetuar as operações.

9. Referências

ALBUQUERQUE ET AL. **The Robot from Ipanema goes Working: Estimating the Probability of Jobs Automation in Brazil**, Latin American Business Review, 20:3, 227-248, DOI: 10.1080/10978526.2019.1633238. Disponível em: <https://www.tandfonline.com/doi/abs/10.1080/10978526.2019.1633238>. Acesso em: 5 abr. 2021.

ALEXOPOULOS, M.; COHEN, J. **THE MEDIUM IS THE MEASURE: TECHNICAL CHANGE AND EMPLOYMENT, 1909–1949**. 1 out. 2016. Disponível em: <https://direct.mit.edu/rest/article/98/4/792/58341/The-Medium-Is-the-Measure-Technical-Change-and>. Acesso em: 24 mai. 2021.

BURNASHEV, A. **Gspread Documentation, Release 3.7.0**. 16 abr. 2021. Disponível em: <https://buildmedia.readthedocs.org/media/pdf/gspread/latest/gspread.pdf>. Acesso em: 11 mai. 2021.

CAIMI, D. **O PROBLEMA DO PIXEL MISTURA: UM ESTUDO COMPARATIVO***. 1993. Disponível em: <https://www.lume.ufrgs.br/bitstream/handle/10183/1373/000106990.pdf>. Acesso em: 24 mai. 2021.

CARR, D. J. **What Value do you Create? Marketing's 3 Types of Value**. 2019. Disponível em: <https://djc1805.medium.com/what-value-do-you-create-marketings-3-types-of-value-7b5f40cad756>. Acesso em: 9 mai. 2021.

CGMA. **Kotler's five product level model**. Disponível em: <https://www.cgma.org/resources/tools/cost-transformation-model/kotlers-five-product-level-model.html>. Acesso em: 7 mai. 2021.

CHAI, J.; JOHAR, F. **A farthest insertion heuristic algorithm for the distance-constrained capacitated vehicle routing problem**. 2015. Disponível em: <https://www.semanticscholar.org/paper/A-farthest-insertion-heuristic-algorithm-for-the-Chai-Johar/8ab60e87a3dbd9bac80da25bc33d422094acf4d7>. Acesso em: 28 abr. 2021.

CHANDRA, R. V., & VARANASI, B. S. **Python requests essentials**. Packt Publishing Ltd, 2015.

DANTZIG, G.B.; RAMSER, J.H. **THE TRUCK DISPATCHING PROBLEM**. Management Science, Vol. 6, No. 1, p. 80-91, out. 1959. Disponível em: <https://andresjaquep.files.wordpress.com/2008/10/2627477-clasico-dantzig.pdf>. Acesso em: 24 mai. 2021.

DE OLIVEIRA, H. C. B. **ALGORITMO ONLINE PARA O PROBLEMA DINÂMICO DE ROTEAMENTO DE VEÍCULOS**. 2011. 147 f. Tese de Doutorado (Ciência da Computação) - Instituto de Ciências Exatas, Universidade Federal de Minas Gerais. Disponível em: <https://repositorio.ufmg.br/bitstream/1843/RVMR-8PQPM2/1/humbertocesar.pdf>. Acesso em: 8 abr. 2021.

DEMING, W. E. **Qualidade: a revolução da administração**. Editora Marques Saraiva, 1990.

DUDA, R. O. & HART, P. E. **Pattern classification and scene analysis.** Vol. 3, pp. 731-739, New York: Wiley, 1973.

FISCHER-BAUM, R. **WHAT 'TECH WORLD' DID YOU GROW UP IN?**. Washington Post. 2017. Disponível em:

<https://www.washingtonpost.com/graphics/2017/entertainment/tech-generations/>. Acesso em: 5 abr. 2021.

FOOT, A. **Are you spending too much on logistics?**. eDesk. Disponível em: <https://blog.edesk.com/resources/are-you-spending-more-than-you-should-on-logistics/>. Acesso em: 7 mai. 2021.

GEETHA, S.; POONTHALIR, G.; VANATHI, P. T. **Improved K-Means Algorithm for Capacitated Clustering Problem**. 2009. PSG College of Technology, Tamil Nadu, India. Disponível em: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.414.2123&rep=rep1&type=pdf>. Acesso em: 9 abr. 2021.

GEOPY CONTRIBUTORS. **GeoPy Documentation, Release 1.23.0**. 04 jul. 2020. Disponível em: <https://geopy.readthedocs.io/_/downloads/en/v1/pdf/>. Acesso em: 11 mai. 2021.

GOULART, A. C. **ANÁLISE E OTIMIZAÇÃO DAS ROTAS DE DISTRIBUIÇÃO DOS PRODUTOS DE UMA EMPRESA**. 2015. 64 f. Trabalho de Conclusão de Curso (Graduação em Engenharia de Produção) - Faculdade de Tecnologia, Universidade de Brasília. Disponível em: <https://www.bdm.unb.br/bitstream/10483/13254/1/2015_AlineCardosoGoulart.pdf>. Acesso em: 24 mar., 2021.

GLOBAL INNOVATION INDEX. **2020 Economy Reports**. Disponível em: <https://www.globalinnovationindex.org/analysis-economy>. Acesso em: 5 abr. 2021.

HARRIS, C. R. ET AL. **Array programming with NumPy**. Nature 585, 357–362 (2020). DOI: 0.1038/s41586-020-2649-2. Disponível em: <https://www.nature.com/articles/s41586-020-2649-2>. Acesso em: 11 mai. 2021.

HART, J. P.; SHOGAN, A. W. **Semi-greedy heuristics: An empirical study** (1987). Disponível em: <https://www.sciencedirect.com/science/article/abs/pii/0167637787900216>. Acesso em: 28 abr. 2021.

HATAMLOU, A.; ABDULLAH, S.; NEZAMABADI-POUR, H. **A combined approach for clustering based on K-means and gravitational search algorithms**. 2012. Islamic Azad University, Department of Electrical Engineering, Khoy Branch, Iran. DOI: https://doi.org/10.1016/j.swevo.2012.02.003. Disponível em: <https://www.sciencedirect.com/science/article/abs/pii/S2210650212000235>. Acesso em: 8 abr. 2021.

HOLST, A. **Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2024** (2021). Disponível em: <https://www.statista.com/statistics/871513/worldwide-data-created/>. Acesso em: 19 abr. 2021.

ILIA. **Millions of Drive-Times (OSM + Python + STATA)**. 2016. Ilia's Data Science Blog. Disponível em: <https://iliauk.wordpress.com/2016/02/23/millions-of-distances-osrm-python/>. Acesso em: 9 abr. 2021.

INTERNET WORLD STATS. **INTERNET GROWTH STATISTICS**. 2021. Disponível em: <https://www.internetworldstats.com/emarketing.htm>. Acesso em: 5 abr. 2021.

JAMES, G. ET AL. **An introduction to statistical learning**. Vol. 112, p. 18, New York: springer, 2013.

KIM, K. **Capacitated Vehicle Routing Problem (CVRP) with Python+Pulp and Google Maps API**. Medium: JDSC Tech Blog. 2020. Disponível em:

<https://medium.com/jdsc-tech-blog/capacitated-vehicle-routing-problem-cvrp-with-python-pulp-and-google-maps-api-5a42dbb594c0>. Acesso em: 14 abr. 2021.

KPMG. **Investment in technology innovation**. 2019. Disponível em: <https://home.kpmg/us/en/home/insights/2019/06/investment-in-technology-innovation.html>. Acesso em: 5 abr. 2021.

KPMG. **Investment in technology innovation: KPMG Technology Industry Innovation Survey**. 2019. Disponível em: <https://assets.kpmg/content/dam/kpmg/us/pdf/2019/06/investment-in-technology-innovation-2019.pdf>. Acesso em: 5 abr. 2021.

LIPOVICH, I. **How To Use Technology To Gain A Sustainable Competitive Advantage**. Forbes Business Council, 2020. Disponível em: <https://www.forbes.com/sites/forbesbusinesscouncil/2020/10/06/how-to-use-technology-to-gain-a-sustainable-competitive-advantage/?sh=25f0a868328c>. Acesso em: 10 mai. 2021.

LUNDH, F. **An introduction to tkinter**. 1999. Disponível em: <www.Pythonware.Com/Library/Tkinter/Introduction/Index>. Acesso em: 11 mai. 2021.

MAURO, P.; MEDAS, P.; FOURNIER, J. **The Cost Of Corruption**. 2019. Disponível em: <https://www.imf.org/external/pubs/ft/fandd/2019/09/pdf/the-true-cost-of-global-corruption-mauro.pdf>. Acesso em: 10 mai. 2021.

MCKINNEY, W. **Data Structures for Statistical Computing in Python**. Proceedings of the 9th Python in Science Conference, p. 56-61. Stefan van der Walt and Jarrod Millman. DOI: 10.25080/Majora-92bf1922-00a

MCKINSEY&COMPANY. **JOBS LOST, JOBS GAINED: WORKFORCE TRANSITIONS IN A TIME OF AUTOMATION**. McKinsey Global Institute, 2017. Disponível em: <https://www.mckinsey.com/~/media/mckinsey/industries/public%20and%20social%20sector/our%20insights/what%20the%20future%20of%20work%20will%20mean%20for%20jobs%20skills%20and%20wages/mgi%20jobs%20lost-jobs%20gained_report_december%202017.pdf>. Acesso em: 25 mar. 2021.

PAPADIMITRIOU, C.H.; STEIGLITZ, K. **Combinatorial Optimization: Algorithms and Complexity**. Jan. 1982. Disponível em: <https://www.researchgate.net/publication/220695898_Combinatorial_Optimization_Algorithms_and_Complexity>. Acesso em: 24 mai. 2021.

PESQUISA FAPESP. **Dispêndios em P&D no estado de São Paulo** (2019). Disponível em: <https://revistapesquisa.fapesp.br/dispendios-em-pd-no-estado-de-sao-paulo/>. Acesso em: 7 abr. 2021.

PINTO, A. A.; DE SOUZA, S. R. **SOLUÇÃO DO PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM JANELA DE TEMPO VIA ITERATED GREEDY SEARCH**. Revista Interdisciplinar de Pesquisa em Engenharia, [S. l.], v. 2, n. 9, p. 182–195, 2017. DOI: 10.26512/ripe.v2i9.15042. Disponível em: <https://periodicos.unb.br/index.php/ripe/article/view/15042>. Acesso em: 24 de mar., 2021.

PIRES, M. **A evolução dos investimentos públicos: 1947-2017**. Observatório de Política Fiscal, Instituto Brasileio de Economia (2018). Disponível em: <https://observatorio-politica-fiscal.ibre.fgv.br/posts/evolucao-dos-investimentos-publicos-1947-2017>. Acesso em: 26 de mar. 2021.

PYTHON-VISUALIZATION. **Folium**, 2020. Disponível em: <https://python-visualization.github.io/folium/>. Acesso em: 11 mai. 2021.

RINGEL ET AL. **In Innovation, Big Is Back: The Most Innovative Companies 2020**. 2020. Boston Consulting Group. Disponível em: <https://www.bcg.com/publications/2020/most-innovative-companies/large-company-innovation-edge>. Acesso em: 5 abr. 2021.

SANTOS, J. C. S. **Como e quando surgiram as primeiras empresas?**. Project Management Knowledge Base, 2017. Disponível em: <http://pmkb.com.br/artigos/como-e-quando-surgiram-as-primeiras-empresas/>. Acesso em: 9 mai. 2021.

SATHER, A. **R&D Spending as a Percentage of Revenue By Industry [S&P500]**. Sather Research, 2021. Disponível em: <https://einvestingforbeginners.com/rd-spending-as-a-percentage-of-revenue-by-industry>. Acesso em: 26 de mar. 2021.

SCARTEZINI, L. M. B. **Análise e Melhoria de Processos**. 2009. Disponível em: <http://siseb.sp.gov.br/arqs/GE%20B%20-%20An%C3%A1lise-e-Melhoria-de-Processos.pdf>. Acesso em: 26 mai. 2021.

SELECTUSA U.S. DEPARTMENT OF COMMERCE. **LOGISTICS AND TRANSPORTATION SPOTLIGHT**. Disponível em: <https://www.selectusa.gov/logistics-and-transportation-industry-united-states>. Acesso em: 7 mai. 2021.

SHEWART, W. A. **Statistical method from the view point of quality control**. 1939.

SKILLICORN, N. **Top 1000 companies that spend the most on Research & Development (charts and analysis)**. Ago. de 2019. Disponível em: <https://www.ideatovalue.com/inno/nickskillicorn/2019/08/top-1000-companies-that-spe

nd-the-most-on-research-development-charts-and-analysis/#fortune500>. Acesso em: 5 abr. 2021.

SOUZA, L. S. **GERENCIAMENTO DE PROCESSOS: PROPOSTA DE MELHORIA DE DESEMPENHO ORGANIZACIONAL DO IFB CAMPUS SAMAMBAIA**. 2016. Disponível em: <https://repositorio.unb.br/bitstream/10482/20840/1/2016_LuidsonSaraivaSouza.pdf>. Acesso em: 26 mai. 2021.

statistics — Mathematical statistics functions. Python Software Foundation, 2021. Disponível em: <https://docs.python.org/3/library/statistics.html>. Acesso em: 11 mai. 2021.

STRATEGY&. **The 2018 Global Innovation 1000 study** (2018). Disponível em: <https://www.strategyand.pwc.com/gx/en/insights/innovation1000.html>. Acesso em: 5 abr. 2021.

THE PANDAS DEVELOPMENT TEAM. **Pandas-dev/pandas: Pandas**. Fev. de 2020. Disponível em: <https://doi.org/10.5281/zenodo.3509134>. Acesso em: 11 mai. 2021.

TIOBE. **TIOBE Index for May 2021**. 2021. Disponível em: <https://www.tiobe.com/tiobe-index/>. Acesso em: 11 mai. 2021.

TRUONG, A. **What tech companies spent on R&D relative to revenue**. 2016. Disponível em: <https://theatlas.com/charts/N1Gs8E4v>. Acesso em: 26 mar. 2021.

UFRJ: Prof. Federico Gálvez-Durand. **LINGUAGEM DE MÁQUINA**. Rio de Janeiro: UFRJ, 1999. Página inicial. Disponível em <http://www.del.ufrj.br/~federico.besnard/del/eel485/ep3/ep3.html>. Acesso em: 25 de mar. 2021.

UNESCO INSTITUTE FOR STATISTICS. **Expenditure on research and development (R&D) as a percentage of GDP** (2021). Disponível em: <http://data.uis.unesco.org/#>. Acesso em: 6 abr. 2021.

VAN ROSSUM, G. **Math, The Python Library Reference, release 3.8.2**. Python Software Foundation, 2020.

VAN ROSSUM, G., & DRAKE, F. L. **Python 3 Reference Manual**. Scotts Valley, CA: CreateSpace, 2009.

WAGNER-TSUKAMOTO, S. **Scientific Management**. 2018. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/9781405165518.wbeos1198>. Acesso em: 8 mai. 2021.

Webbrowser: Convenient Web-browser controller. Python Software Foundation, 2021. Disponível em: https://docs.python.org/3/library/webbrowser.html. Acesso em: 11 mai. 2021.

WOODS, M. **Gmplot: Plotting data on Google Maps, the easy (stupid) way**. 2017. Disponível em: https://Github.Com/Gmplot/Gmplot. Acesso em: 11 mai. 2021.

ZACKS INVESTMENT RESEARCH. **R&D Spending Hits the Fastest Pace in 12 Years: Top 5 Winners**. NASDAQ, 2018. Disponível em: <https://www.nasdaq.com/articles/rd-spending-hits-fastest-pace-12-years-top-5-winners-2018-08-24>. Acesso em: 26 de mar. 2021.

10. Apêndice

10.1 Resolvedor de Roteamento para Veículos

Este projeto disponibiliza publicamente um software para roteamento de um veículo. A resolução conhecida como Problema do Caixeiro Viajante consiste na reorganização de um dado conjunto de endereços, de forma que a ordem retornada minimize o tamanho do trajeto e o tempo em trânsito. O software oferecido contém uma interface para a fácil resolução da otimização de trajetos: Você escreve os endereços inicial e final, e um conjunto de endereços intermediários, e o programa retorna a ordem ótima entre eles. É útil para economizar gasolina e tempo na locomoção urbana, e otimizar custos de operações ou viagens que utilizam deslocamento automotivo.

Para obter a versão atualizada, acesse o link e siga as instruções.

https://github.com/victorccaldas/tsp_pcv_vrotas

Alternativamente, se souber o que está fazendo, você pode copiar o código abaixo e executá-lo através de um Interpretador Python. Você também pode modificar o código conforme suas necessidades, e sugerir ou incluir mudanças através do repositório do programa no link acima.

10.1.1 Código aberto

vrotas_tsp_pcv.py v1.0.0

```
from tkinter.scrolledtext import ScrolledText
from geopy.geocoders import Nominatim
from datetime import timedelta
from datetime import datetime
from tkinter import *
import webbrowser
import googlemaps
import requests
import polyline
import folium
import sys
```

```
chrome_path = 'C:/Program Files (x86)/Google/Chrome/Application/chrome.exe %s'

geolocator = Nominatim(user_agent="enderecos_routing")

google_key = ''

cor_dos_labels = 'white'

def aplicarRotas():
    global lista_enderecos
    global obs_google

    lista_enderecos = end_clientes.get(1.0, END).split("\n")

    while '' in lista_enderecos:
        lista_enderecos.remove('')

    try:
        canvas.delete(vermapa1_button)
    except:
        pass
    try:
        canvas.delete(vermapa2_button)
    except:
        pass
    try:
        canvas.delete(obs_google)
    except:
        pass

    if servico_escolhido.get() == "Google":
        servicoRotas_label.config(bg='#d4d0cf', text=servico_escolhido.get())
        obs_google = canvas.create_window(460, 440, window=Label(root_ends_routing,
bg='#d4d0cf',text="Obs: o Google otimiza\n no máximo 23 endereços\ne desenha no máx.\n10 caminhos",
font=('helvetica', 8)))

        try:
            G_routing()
        except:
            print("Deu errado")
            canvas.delete(distance_notoptimal_label)
            canvas.delete(duration_notoptimal_label)
            canvas.delete(distance_optimal_label)
            canvas.delete(duration_optimal_label)

    elif servico_escolhido.get() == "OSRM":
        try: canvas.delete(obs_google)
        except: pass
        servicoRotas_label.config(bg='#d4d0cf', text=servico_escolhido.get())
        otimização = OSRM_routing('otimizar',lista_enderecos, endereco_inicio.get(), endereco_fim.get())
```

```
        if type(otimização) == list:
            resultado_otimizado.config(bg='#d39994') # light red
            resultado_otimizado.delete(1.0,END)
            resultado_otimizado.insert(END,'Endereços inválidos:\n\n'+'\n'.join(otimização))
            return

        duracao = otimização['duration']/60
        distancia = otimização['distance']/1000
        enderecos_otimizados = [lista_enderecos[int(x)] for x in otimização['best_order']]

        não_otimização = OSRM_routing('nao_otimizar',lista_enderecos, endereco_inicio.get(),
endereco_fim.get())
        duracao_ñotima = não_otimização['duration']/60
        distancia_ñotima = não_otimização['distance']/1000

        distance_optimal_label = canvas.create_window(600, 430, window=Label(root_ends_routing,
bg='#d3e6d3', text=("Distância: "+str("%.1f" % distancia)+" km"), anchor="w", font=('helvetica', 10)))
        duration_optimal_label = canvas.create_window(600, 455, window=Label(root_ends_routing,
bg='#d3e6d3', text=("Duração: "+str("%.1f" % duracao)+" min"), anchor="w", font=('helvetica', 10)))

        distance_notoptimal_label = canvas.create_window(90, 430, window=Label(root_ends_routing,
bg='#e8d8b7', text=("Distância: "+str("%.1f" % distancia_ñotima)+" km"), anchor='w', font=('helvetica',
10)))
        duration_notoptimal_label = canvas.create_window(90, 455, window=Label(root_ends_routing,
bg='#e8d8b7', text=("Duração: "+str(("%.1f" % duracao_ñotima)+" min")), anchor='w', font=('helvetica',
10)))

        # inserir outro text box com enderecos_otimizados
        resultado_otimizado.delete(1.0,END)
        resultado_otimizado.insert(END,'\n'.join(enderecos_otimizados))
        resultado_otimizado.config(bg='white')

        # Botões de ver as rotas (mapa1.html e mapa2.html)
        vermapa1_button = canvas.create_window(260, 440, window=Button(text="Ver rota antes",
width=15,command=lambda:verRotaNoMapa('mapa1'), bg='#579482', fg='white', font=('helvetica', 13, 'bold')))
        vermapa2_button = canvas.create_window(770, 440, window=Button(text="Ver rota depois",
width=15,command=lambda:verRotaNoMapa('mapa2'), bg='#579482', fg='white', font=('helvetica', 13, 'bold')))

def Waze_routing():
    start_address = endereco_inicio.get()
    end_address = endereco_fim.get()
    region = 'SA'
    route = WazeRouteCalculator.WazeRouteCalculator(start_address, end_address, region)
    route.calc_route_info()

def get_map_folium(melhor_rota, coords_clientes, nome_mapa):
    # criando mapa
    mapa = folium.Map(location=[(melhor_rota['start_point'][0] + melhor_rota['end_point'][0])/2,
                                (melhor_rota['start_point'][1] + melhor_rota['end_point'][1])/2],
                      zoom_start=12)
```

```
    # traçando caminho
    folium.PolyLine(melhor_rota['route'], weight=8, color='blue', opacity=0.5).add_to(mapa)

    # plotando inicio e fim
    folium.Marker(location=melhor_rota['start_point'], tooltip="Início", icon=folium.Icon(icon='play',
color='green')).add_to(mapa)
    folium.Marker(location=melhor_rota['end_point'],tooltip="Final", icon=folium.Icon(icon='stop',
color='red')).add_to(mapa)

    # plotando clientes
    contagem = 0
    for coordenada in coords_clientes:
        contagem +=1
        lat = coordenada.split(',')[0]
        lng = coordenada.split(',')[1]
        folium.Marker(location=(lat, lng), popup="Este é o "+str(contagem)+"º waypoint",
tooltip=contagem).add_to(mapa)
    mapa.save(nome_mapa+".html")

def osrm_obter_trajeto_nao_otimo(lista_waypoints, coordenada_inicio, coordenada_final):
    # invertendo ordem das coordenadas inicio e fim
    coord_inicial_lng_lat =
(coordenada_inicio.split(",")[1]+','+coordenada_inicio.split(",")[0]).replace(' ','')
    coord_final_lng_lat = (coordenada_final.split(",")[1]+','+coordenada_final.split(",")[0]).replace('
','')

    wps_lng_lat = [(i.split(',')[1]+","+i.split(',')[0]).replace(" ","") for i in lista_waypoints]

    # transformando waypoints em uma string unica , e adicionando coord inicial e final
    wp_string = coord_inicial_lng_lat+';'
    for i in wps_lng_lat:
        wp_string += i+';'
    wp_string += coord_final_lng_lat

    # Resultados ROUTE
    route_url =
'http://localhost:5000/route/v1/driving/{}?steps=true&annotations=true&continue_straight=false'.format(wp_
string)
    print(route_url)
    route_results = requests.get(url=route_url)
    route_data = route_results.json()
    polylines_list = polyline.decode(route_data['routes'][0]['geometry'])

    # organizando resultados
    rota = {'polyline':route_data['routes'][0]['geometry'],
                    'route': polylines_list,
                    'legs':route_data['routes'][0]['legs'],
                    'start_point': [route_data['waypoints'][0]['location'][1],
route_data['waypoints'][0]['location'][0]],
                    'end_point': [route_data['waypoints'][-1]['location'][1],
route_data['waypoints'][-1]['location'][0]],
```

```
                    'duration':route_data['routes'][0]['duration'],
                    'distance':route_data['routes'][0]['distance']}

    get_map_folium(rota, lista_waypoints, 'mapa1')

    return rota

def osrm_obter_trajeto(lista_waypoints,coordenada_inicio, coordenada_final):
    # invertendo ordem das coordenadas inicio e fim
    coord_inicial_lng_lat =
(coordenada_inicio.split(",")[1]+','+coordenada_inicio.split(",")[0]).replace(' ','')
    coord_final_lng_lat = (coordenada_final.split(",")[1]+','+coordenada_final.split(",")[0]).replace('
','')

    # eliminando duplicatas e invertendo ordem dos waypoints
    #waypoints_sem_duplicatadas = list(dict.fromkeys(lista_waypoints))
    #wps_lng_lat = [(i.split(',')[1]+","+i.split(',')[0]).replace(" ","") for i in
waypoints_sem_duplicatadas]
    wps_lng_lat = [(i.split(',')[1]+","+i.split(',')[0]).replace(" ","") for i in lista_waypoints]

    # transformando waypoints em uma string unica , e adicionando coord inicial e final
    wp_string = coord_inicial_lng_lat+';'
    for i in wps_lng_lat:
        wp_string += i+';'
    wp_string += coord_final_lng_lat
    # Resultados TRIP
    trip_url =
'http://localhost:5000/trip/v1/driving/{}?roundtrip=false&source=first&destination=last&steps=true&annotat
ions=true&geometries=polyline&overview=simplified'.format(wp_string)
    #print(trip_url)
    trip_results = requests.get(url=trip_url)
    trip_data = trip_results.json()
    polylines_list = polyline.decode(trip_data['trips'][0]['geometry'])

    # devem ser visitados no formato otimizado
    recommended_order = [i['waypoint_index']-1 for i in trip_data['waypoints'][1:-1]]
    lista_reordenando = [(x,recommended_order.index(x)) for x in recommended_order]
    lista_reordenando.sort()
    waypoints_optimal_order = [x[1] for x in lista_reordenando]

    # organizando resultados
    melhor_rota = {'polyline':trip_data['trips'][0]['geometry'],
                    'route': polylines_list,
                    'legs':trip_data['trips'][0]['legs'],
                    'start_point': [trip_data['waypoints'][0]['location'][1],
trip_data['waypoints'][0]['location'][0]],
                    'end_point': [trip_data['waypoints'][-1]['location'][1],
trip_data['waypoints'][-1]['location'][0]],
                    'best_order': waypoints_optimal_order,
                    'duration':trip_data['trips'][0]['duration'],
                    'distance':trip_data['trips'][0]['distance']}
```

```
    coords_otimizadas = [lista_waypoints[int(x)] for x in melhor_rota['best_order']]
    get_map_folium(melhor_rota, coords_otimizadas, 'mapa2')

    return melhor_rota

def achar_coordenada(endereco):
    import time
    global resultado
    global achar_coordenada_servico_utilizado

    if str(endereco) == 'nan' or str(endereco) == '':
        return 'Não encontrada'

    # 1) Primeiro tentar nominatim
    def meu_geolocator(client, endereco):
        return client.geocode(endereco)

    retrys = 0
    while True:
        try:
            resultado = meu_geolocator(geolocator,endereco)

            break
        except Exception as erro:
            print("Tentativas: "+str(retrys)+"\nErro: \n"+str(erro)+'\n')
            retrys += 1
            time.sleep(1)
            pass
        if retrys > 5:
            resultado = None
            break

    if resultado != None:
        resultado = (resultado.latitude,resultado.longitude)

        achar_coordenada_servico_utilizado = "Nominatim"

        LAT = resultado[0]
        LNG = resultado[1]
        if -17 < LAT < -14 and -49 < LNG < -46:
            resultado = str(LAT)+","+str(LNG) #str(resultado).replace(')','').replace('(','')
            return resultado
        else:
            resultado = "Inválida: "+str(resultado)

    # 2) Se ñ encontrada ou fora de Brasília, tentar Google
    if type(resultado) == str or resultado == None:
        while True:
            try:
                resultado = meu_geolocator(gmaps_client,endereco)
```

```
                break
            except Exception as erro:
                print("Tentativas: "+str(retrys)+"\nErro: \n"+str(erro)+'\n')
                retrys += 1
                time.sleep(1)
                pass
            if retrys > 5:
                resultado = []
                break

        if resultado != []:
            resultado = (resultado[0]['geometry']['location']['lat'],
resultado[0]['geometry']['location']['lng'])
            achar_coordenada_servico_utilizado = "Google"

            LAT = resultado[0]
            LNG = resultado[1]
            if -17 < LAT < -14 and -49 < LNG < -46:
                resultado = str(LAT)+","+str(LNG)
                return resultado
            else:
                resultado = "Inválida: "+str(resultado)

    if type(resultado) == str or resultado == [] or resultado == None:
        resultado = 'Não encontrada'
        achar_coordenada_servico_utilizado = 'nenhum'
    else:
        resultado = "Bugou em algum lugar..."
    return resultado

def geocode_osm(endereços):
    coords_intermediarias = []
    ends_invalidos = []
    for end in endereços:
        try:
            coord_end = str(geolocator.geocode(end).latitude)+','+str(geolocator.geocode(end).longitude)
            coords_intermediarias.append(coord_end)
        except:
            ends_invalidos.append(end)
    return coords_intermediarias, ends_invalidos

def geocode_osm_google(endereços):
    coords_intermediarias = []
    ends_invalidos = []
    for end in endereços:
        try:
            coord_end = achar_coordenada(end)
            if coord_end != 'Não encontrada':
                coords_intermediarias.append(coord_end)
            else:
               ends_invalidos.append(end)
```

```
        except:
            ends_invalidos.append(end)
    return coords_intermediarias, ends_invalidos

def OSRM_routing(otimizar,intermediarios, inicio, fim):
    coord_inicial = str(geolocator.geocode(inicio).latitude)+','+str(geolocator.geocode(inicio).longitude)
    coord_final = str(geolocator.geocode(fim).latitude)+','+str(geolocator.geocode(fim).longitude)

    if google_key == '':
        coords_intermediarias, ends_invalidos = geocode_osm(intermediarios)
    else:
        coords_intermediarias, ends_invalidos = geocode_osm_google(intermediarios)

    if len(ends_invalidos) > 0:
        print("Endereços inválidos:\n")
        [print(end) for end in ends_invalidos]
        return ends_invalidos
    else:
        if otimizar == 'otimizar':
            melhor_rota = osrm_obter_trajeto(coords_intermediarias, coord_inicial, coord_final)
            return melhor_rota
        else:
            rota = osrm_obter_trajeto_nao_otimo(coords_intermediarias, coord_inicial, coord_final)
            return rota

def G_routing():
    global coords_otimizado
    global nova_rota
    global distance_notoptimal_label
    global duration_notoptimal_label
    global distance_optimal_label
    global duration_optimal_label
    global enderecoslidos
    global start_address
    global vermapa1_button
    global vermapa2_button

    try:
        horario_string = horario_inicio.get()
        horario_datetime = datetime.strptime(horario_string, '%d/%m/%y %H:%M')

    except:
        print("A data está no formato errado.")
        enderecoerrado_label.delete('1.0',END)
        enderecoerrado_label.insert(END,horario_string)
        enderecoerrado_label.configure(bg='#bf8e88')
        return 'stop'

    if (datetime.now() - horario_datetime).days > 0:
        print("Erro: A data está no passado.")
        enderecoerrado_label.delete('1.0',END)
```

```
        enderecoerrado_label.insert(END,horario_string)
        enderecoerrado_label.configure(bg='#bf8e88')
        return 'stop'
    #location = gmaps_client.geocode(verificacao)

    lista_coords = []
    bsb_geocode = gmaps_client.geocode("Brasilia,DF")
    bsb_coords = (bsb_geocode[0]['geometry']['location']['lat'],
bsb_geocode[0]['geometry']['location']['lng'])
    start_address = endereco_inicio.get()
    start_geocode = gmaps_client.geocode(start_address)
    end_address = endereco_fim.get()
    end_geocode = gmaps_client.geocode(end_address)

    try:
        start_coords =
(start_geocode[0]['geometry']['location']['lat'],start_geocode[0]['geometry']['location']['lng'])
        enderecoerrado_label.delete('1.0',END)
        enderecoerrado_label.insert(END,'<Debug>')
        enderecoerrado_label.configure(bg='white')
    except:
        enderecoerrado_label.delete('1.0',END)
        enderecoerrado_label.insert(END,start_address)
        enderecoerrado_label.configure(bg='#bf8e88')
        sys.exit()

    try:
        end_coords =
(end_geocode[0]['geometry']['location']['lat'],end_geocode[0]['geometry']['location']['lng'])
        enderecoerrado_label.configure(bg='white')
        enderecoerrado_label.delete('1.0',END)
        enderecoerrado_label.insert(END,'<Debug>')

    except:
        enderecoerrado_label.configure(bg='#bf8e88')
        enderecoerrado_label.delete('1.0',END)
        enderecoerrado_label.insert(END,end_address)
        sys.exit()

    end_coords =
(end_geocode[0]['geometry']['location']['lat'],end_geocode[0]['geometry']['location']['lng'])

    # criar listas de coordenadas, latitude e longitude com base nos endereços escolhidos
    enderecos_lidos = []
    try:
        for endereco in lista_enderecos:
            location = gmaps_client.geocode(endereco)
            #print(location.address)
            #print((location.latitude, location.longitude))
            lista_coords.append((location[0]['geometry']['location']['lat'],
location[0]['geometry']['location']['lng']))
```

```
            enderecos_lidos.append(endereco)
    except:
        for end in lista_enderecos:
            if end in enderecos_lidos:
                pass
            else:
                print(end)
                enderecoerrado_label.configure(bg='#bf8e88')
                enderecoerrado_label.delete('1.0',END)
                enderecoerrado_label.insert(END,end)
                sys.exit()

# Rota NÃO otimizada : tempo e distância
    global directions_NaoOtimizado
    directions_NaoOtimizado = gmaps_client.directions(start_coords, end_coords, waypoints=lista_coords,
departure_time=horario_datetime)
    distance_notoptimal = 0
    duration_notoptimal = 0
    legs = directions_NaoOtimizado[0]['legs']
    for leg in legs:
        distance_notoptimal = distance_notoptimal + leg['distance']['value']
        duration_notoptimal = duration_notoptimal + leg['duration']['value']

    try:
        canvas.delete(distance_notoptimal_label)
        canvas.delete(duration_notoptimal_label)
    except:
        pass

    distance_notoptimal_label = canvas.create_window(90, 430, window=Label(root_ends_routing,
bg='#e8d8b7', text=("Distância: "+str("%.1f" % (distance_notoptimal/1000))+" km"), anchor='w',
font=('helvetica', 10)))
    duration_notoptimal_label = canvas.create_window(90, 455, window=Label(root_ends_routing,
bg='#e8d8b7', text=("Duração: "+str(("%.1f" % (duration_notoptimal/60))+" min")), anchor='w',
font=('helvetica', 10)))

    directions_result = gmaps_client.directions(start_coords, end_coords, waypoints=lista_coords,
optimize_waypoints = True, traffic_model = 'best_guess', departure_time=horario_datetime)

    # somando distancias e durações de cada etapa da rota
    distance = 0
    duration = 0
    legs = directions_result[0]['legs']

    for leg in legs:
        #distance = distance + leg.get("distance").get("value") # ambas as formas dão no mesmo
        distance = distance + legs[0]['distance']['value']
        duration = duration + legs[0]['duration']['value']

    # mostrar distância, tempo e ordem otimizada da rota
    print(str("%.1f" % (distance/1000))+" km")
```

```
    print(str(duration/60)+" min")
    print(directions_result[0]['waypoint_order'])

    # lista com endereços e coordenadas reorganizados no formato otimizado
    nova_rota = []
    coords_otimizado = []
    for wp_num in directions_result[0]['waypoint_order']:
        nova_rota.append(lista_enderecos[wp_num])
        coords_otimizado.append(lista_coords[wp_num])

    # transformando enderecos de lista pra string
    enderecos_otimizados = '\n'.join(nova_rota)

    # criar plotter, criar direções e plotar no mapa (mapa1)
    plotmap = gmplot.GoogleMapPlotter(bsb_coords[0],bsb_coords[1], 13, apikey=g_apikey)
    plotmap.directions(start_coords, end_coords, waypoints=lista_coords, optimize_waypoints = True,
traffic_model = 'best_guess',departure_time=horario_datetime)

    # Plotando enderecos (markers) na ordem não-otimizada
    marker_label = 1
    plotmap.marker(start_coords[0],start_coords[1],precision=10,
label=str(marker_label),title=str(marker_label))
    marker_label += 1
    for i in lista_coords:
        plotmap.marker(i[0],i[1],precision=10, label=str(marker_label),title=str(marker_label))
        marker_label += 1
    plotmap.marker(end_coords[0],end_coords[1],precision=10,
label=str(marker_label),title=str(marker_label))

    plotmap.draw('mapa1.html')
    global duracao_otima
    global duracao_rota
    global dict_rotas

    # Plottando mapa2.html
    distancia_otima = 0
    duracao_otima = 0
    marker_label = 1
    anterior = start_coords
    enderecos_rotalidos = []
    dict_rotas = {}
    duracao_rota = 0
    duracao_total_da_rota = 0
    todosends = []

    plotmap = gmplot.GoogleMapPlotter(bsb_coords[0],bsb_coords[1], 13, apikey=g_apikey)
    plotmap.marker(start_coords[0],start_coords[1],precision=10,
label=str(marker_label),title=start_address, color='indianred',draggable=True)
    marker_label += 1
```

```
# PENSAR EM UMA FORMA de fazer o datetime.now() calcular corretamente, visto que todos os pontos do
trajeto estarão
# com departure no horário mencionado no user_input 'horario de inicio'

    for i in coords_otimizado:
        plotmap.directions(anterior, i, traffic_model ='best_guess', departure_time=horario_datetime)
        plotmap.marker(i[0],i[1],precision=10, label=str(marker_label),title=nova_rota[(marker_label-2)],
color='indianred',draggable=True)
        caminho_1po1 = gmaps_client.directions(anterior, i, traffic_model ='best_guess',
departure_time=horario_datetime)
        distancia_otima = distancia_otima + caminho_1po1[0]['legs'][0]['distance']['value']
        duracao_otima = duracao_otima + caminho_1po1[0]['legs'][0]['duration']['value']

        marker_label += 1
        anterior = i

        duracao_rota = duracao_rota + caminho_1po1[0]['legs'][0]['duration']['value']
        enderecos_rotalidos.append(i)
        duracao_total_da_rota = (len(enderecos_rotalidos)*600) + duracao_rota # qtd de ends dos clientes
lidos multiplicados por 2000 (20 min) + duracao do trajeto

        if duracao_total_da_rota > 6000: # 6000s = 100min (1h40); 50 minutos de folga do final. máx =
140min
            dict_rotas[(duracao_total_da_rota)] = enderecos_rotalidos
            enderecos_rotalidos = []
            duracao_rota = 0

    #ctypes.windll.user32.MessageBoxW(0, dict_rotas, 'Rotas otimizadas:', 1)
    plotmap.directions(anterior, end_coords, traffic_model ='best_guess', departure_time=horario_datetime)
    plotmap.marker(end_coords[0],end_coords[1],precision=10, label=str(marker_label),title=end_address,
color='indianred',draggable=True)
    # Ñ preciso incluir o tempo do ultimo endereço até o final na duracao_rota, já q não influencia no
tempo pros clientes
    caminho_1po1 = gmaps_client.directions(anterior, end_coords, traffic_model ='best_guess',
departure_time=horario_datetime)
    ultimoend_dur = caminho_1po1[0]['legs'][0]['duration']['value']
    distancia_otima = distancia_otima + caminho_1po1[0]['legs'][0]['distance']['value']
    duracao_otima = duracao_otima + caminho_1po1[0]['legs'][0]['duration']['value']

    duracao_total_da_rota = (len(enderecos_rotalidos)*600) + duracao_rota # 1200 = 20 min
    if (duracao_total_da_rota) > 0:
        dict_rotas[(duracao_total_da_rota)] = enderecos_rotalidos

    somadict = 0
    for i in dict_rotas:
        somadict = somadict +i

    print("Resultado = "+str(len(coords_otimizado)*10)+" + "+str((duracao_otima/60)-(ultimoend_dur/60))+"
== "+str(somadict/60))
    # (duracao_otima/60)-(ultimoend_dur/60) tem que ser igual à soma do dict - (enderecos*20)
```

```
    print(dict_rotas)

    try:
        canvas.delete(distance_optimal_label)
        canvas.delete(duration_optimal_label)
    except:
        pass

    distance_optimal_label = canvas.create_window(600, 430, window=Label(root_ends_routing, bg='#d3e6d3',
text=("Distância: "+str("%.1f" % (distancia_otima/1000))+" km"), anchor="w", font=('helvetica', 10)))
    duration_optimal_label = canvas.create_window(600, 455, window=Label(root_ends_routing, bg='#d3e6d3',
text=("Duração: "+str("%.1f" % (duracao_otima/60))+" min"), anchor="w", font=('helvetica', 10)))

    plotmap.draw('mapa2.html')

    # Removendo Markers A, B do mapa:
    arquivo_mapa = open(os.path.dirname(__file__)+'/deps/mapa2.html').read()
    if 'suppress' not in arquivo_mapa:
        replaced = arquivo_mapa.replace('map: map', 'map: map, suppressMarkers: true')
        writer = open(os.path.dirname(__file__)+'\deps\mapa2.html','w')
        writer.write(replaced)
        writer.close()

    # inserir outro text box com enderecos_otimizados
    resultado_otimizado.delete(1.0,END)
    resultado_otimizado.insert(END,enderecos_otimizados)
    resultado_otimizado.config(bg='white')

    # Botões de ver as rotas (mapa1.html e mapa2.html)
    vermapa1_button = canvas.create_window(260, 440, window=Button(text="Ver rota antes",
width=15,command=lambda:verRotaNoMapa('mapa1'), bg='#229399', fg='white', font=('helvetica', 13, 'bold')))
    vermapa2_button = canvas.create_window(770, 440, window=Button(text="Ver rota depois",
width=15,command=lambda:verRotaNoMapa('mapa2'), bg='#229399', fg='white', font=('helvetica', 13, 'bold')))



def verEnderecoNoMapa(tup_lat_lng, endereço):
    plotmap = folium.Map(location=tup_lat_lng, zoom_start=12)
    folium.Marker(location=tup_lat_lng, popup="Coordenada "+str(tup_lat_lng)+"\n\n"+str(endereço),
tooltip=endereço).add_to(plotmap)
    plotmap.save("map_verloc_individual_folium.html")
    webbrowser.get(chrome_path).open(os.path.dirname(__file__)+"\map_verloc_individual_folium.html")

def verRotaNoMapa(mapanum):
    webbrowser.get(chrome_path).open(os.path.dirname(__file__)+"\{}.html".format(mapanum))

def aplicarVerificacao(tupla_solicitada):
    global verif_result_label

    def criarEntryResultados():
        global verif_result_label
```

```
        global canvas
        verif_result_label = Entry(root_ends_routing, width=25, text=verif_result_var.get(),
font=('helvetica', 9))
        canvas.create_window(355, 80, window=verif_result_label)

    try: verif_result_label.get()
    except:
        criarEntryResultados()

    if tupla_solicitada.count(1) == 0:
        checkboxes_label.config(bg='#d4d0cf', relief="solid",text="Selecione um serviço")

    elif tupla_solicitada.count(1) > 1:
        #checkboxes_label.config(bg='#d4d0cf', relief="solid",text="Marque apenas uma opção")
        checkboxes_label.config(bg='#d4d0cf', relief="solid",text="OSRM e Google")
        verificarEndereco_ambos()


    elif tupla_solicitada[0] == 1:
        checkboxes_label.config(bg='#d4d0cf', relief="solid",text="Open Street Map")
        verificarEndereco_openstreetmap()

    else:
        checkboxes_label.config(bg='#d4d0cf', relief="solid",text="Google Maps")
        verificarEndereco_google()

def verificarEndereco_ambos():
    global verificar_coords
    global vernomapa_button
    global gmaps_client

    if google_key == "":
        verif_result_label.delete(0,END)
        verif_result_label.insert(0, 'Serviço Google indisponível. Insira uma chave de API')
        return
    verificarEndereco_openstreetmap()
    if verif_result_label.get() == 'Endereço não encontrado.':
        verificarEndereco_google()

def verificarEndereco_openstreetmap():
    global vernomapa_button
    global verificar_coords

    verificacao = verificar_end.get()
    try:
        canvas.delete(vernomapa_button)
    except:
        pass

    try:
        location = geolocator.geocode(verificacao)
```

```
        verificar_coords = (location.latitude, location.longitude)
        verificar_lat = location.latitude
        verificar_long = location.longitude
        verif_result_label.config(bg='#d2e8ae', width=45, relief="solid")
        verif_result_label.delete(0,END)
        verif_result_label.insert(0, str('Local encontrado: '+str(verificar_coords)+' '))
        #verif_result_var.set(str('Endereço existe! '+str(verificar_coords)+' '))
        vernomapa_button = canvas.create_window(545, 45, window=Button(text="Ver no mapa",
width=15,command=lambda:verEnderecoNoMapa(verificar_coords, verificacao), bg='#229399', fg='white',
font=('helvetica', 12, 'bold')))

    except:
        verificar_lat = 0
        verificar_long = 0
        verif_result_label.config(bg='#c98f8b', width=25, relief="solid")
        verif_result_label.delete(0,END)
        verif_result_label.insert(0, 'Endereço não encontrado.')
        print(sys.exc_info())

def verificarEndereco_google():
    global verificar_coords
    global vernomapa_button

    if google_key == "":
        verif_result_label.delete(0,END)
        verif_result_label.insert(0, 'Serviço Google indisponível. Insira uma chave de API')
        return
    else:
        gmaps_client = googlemaps.Client(key=google_key)

    verificacao = verificar_end.get()

    try:
        canvas.delete(vernomapa_button)
    except:
        pass

    try:
        location = gmaps_client.geocode(verificacao)
        verificar_coords = (location[0]['geometry']['location']['lat'],
location[0]['geometry']['location']['lng'])
        verif_result_label.config(bg='#d2e8ae', width=45, relief="solid")
        verif_result_label.delete(0,END)
        verif_result_label.insert(0, str('Local encontrado: '+str(verificar_coords)+' '))
        vernomapa_button = canvas.create_window(555, 45, window=Button(text="Ver no mapa",
width=15,command=lambda:verEnderecoNoMapa(), bg='#229399', fg='white', font=('helvetica', 12, 'bold')))

    except:
        verificar_lat = 0
        verificar_long = 0
        verif_result_label.config(bg='#c98f8b', width=25, relief="solid")
```

```
        verif_result_label.delete(0,END)
        verif_result_label.insert(0, 'Endereço não encontrado.')
        print(sys.exc_info())

def inserirApiGoogle():
    global api_label
    root_api_google = Toplevel()
    root_api_google.title('Inserir API do Google')
    canvas2 = Canvas(root_api_google, height = 150, width = 500, bg = '#bad5e8', relief = 'raised')
    canvas2.pack()

    api_atual_label = canvas2.create_window(70, 25, window=Label(root_api_google, text='API inserido:', bg
= '#bad5e8', justify=LEFT, font=('helvetica', 8,'bold')))

    if google_key != '':
        api_label = canvas2.create_window(250, 25, window=Label(root_api_google, bg=cor_dos_labels,
justify=LEFT, text=google_key, font=('helvetica', 8,'bold')))
    else:
        api_label = canvas2.create_window(250, 25, window=Label(root_api_google, bg=cor_dos_labels,
justify=LEFT, text='Nenhuma chave encontrada.', font=('helvetica', 8,'bold')))

    key_input = Entry(root_api_google,width=70, bg='#f5f0f0')
    key_input.insert(END,google_key)
    canvas2.create_window(250, 60, window=key_input)

    def alterar():
        global link
        global api_label
        global google_key
        global servicoGoogle_ativo
        global gmaps_client

        try: canvas2.delete(api_label)
        except: pass

        try:
            # executar um teste com a key antes de atribuir à var google_key
            gmaps_client = googlemaps.Client(key=key_input.get())
            gmaps_client.geocode('Unb, Brasília, DF')
            google_key = key_input.get()
            api_label = canvas2.create_window(250, 25, window=Label(root_api_google, bg='#dcf4e4',
justify=LEFT, text=google_key, font=('helvetica', 8,'bold')))
            servicoGoogle_ativo = canvas.create_window(460, 420, window=Label(root_ends_routing,
bg='#dcf4e4',text='Geocode Google ativo', font=('helvetica', 9)))
            google_checkbox_var.set(1)
            google_checkbox.config(state=NORMAL)

        except googlemaps.exceptions.ApiError:
            api_label = canvas2.create_window(250, 25, window=Label(root_api_google, bg='#c98d79',
justify=LEFT, text="Chave API não tem autorização para ser acessada", font=('helvetica', 8,'bold')))
        except:
```

```
            api_label = canvas2.create_window(250, 25, window=Label(root_api_google, bg='#c98d79',
justify=LEFT, text="Chave API não encontrada", font=('helvetica', 8,'bold')))

    def desativar():
        global api_label
        global google_key
        global servicoGoogle_ativo
        google_key = ''

        try: canvas2.delete(api_label); canvas.delete(servicoGoogle_ativo)
        except: pass

        api_label = canvas2.create_window(250, 25, window=Label(root_api_google, bg='white', justify=LEFT,
text='Nenhuma chave ativa.', font=('helvetica', 8,'bold')))
        google_checkbox_var.set(0)
        google_checkbox.config(state=DISABLED)
    alterar_api_button = Button(root_api_google,text="Alterar API", height=1,
width=12,command=lambda:alterar(),wraplength=90, bg='#9ebaa7', font=('helvetica', 10, 'bold'))
    canvas2.create_window(250, 95, window=alterar_api_button)

    desativar_api_button = Button(root_api_google,text="Desativar API", height=1,
width=12,command=lambda:desativar(),wraplength=90, bg='#9ebaa7', font=('helvetica', 10, 'bold'))
    canvas2.create_window(250, 130, window=desativar_api_button)

root_ends_routing = Tk()
root_ends_routing.title('vRotas - TSP / PCV Solver')
#root_ends_routing.iconbitmap('C:/Users/Victor/Desktop/market ocr/icone ricardinho.gif')

#Main window
canvas = Canvas(root_ends_routing, height = 500, width = 1000, bg = "white", relief = 'raised')
canvas.pack()

# Logo vRotas
logo_vrotas = PhotoImage(file = os.path.dirname(__file__)+"\deps\logo vRotas 9.0 mini.png")
logo_v_label = Label(root_ends_routing, image=logo_vrotas, bg='white')
logo_v_label.place(x=850, y=-5)

# Ícone
root_ends_routing.iconphoto(True, PhotoImage(file=os.path.dirname(__file__)+'\deps\icone vRotas 9.0
mini.png'))
# faiô

# checkboxes
servico_busca_selecionado = 'nenhum'
osm_checkbox_var = IntVar(value=1)
osm_checkbox = Checkbutton(root_ends_routing, text="Open Street Map", variable=osm_checkbox_var)
canvas.create_window(340, 10, window=osm_checkbox)
google_checkbox_var = IntVar(value=0)
google_checkbox = Checkbutton(root_ends_routing, text="Google", variable=google_checkbox_var,
state=DISABLED)
canvas.create_window(440, 10, window=google_checkbox)
```

```
chaveapi_google = Button(root_ends_routing,text="Inserir chave API
Google",width=20,command=lambda:inserirApiGoogle(), bg='#bad5e8', font=('helvetica', 9, 'bold'))
canvas.create_window(580, 10, window=chaveapi_google)

# checkbox labels
checkboxes_label_var = StringVar(root_ends_routing)
checkboxes_label_var.set('')
checkboxes_label = Label(root_ends_routing, text=checkboxes_label_var.get(), font=('helvetica', 9))
canvas.create_window(140, 70, window=checkboxes_label)

# verificar se endereço existe
verificar_label = Label(root_ends_routing, bg=cor_dos_labels, text='Checar coordenadas de endereços:',
font=('helvetica', 11, 'bold'))
canvas.create_window(140, 10, window=verificar_label)

verificar_end = Entry(root_ends_routing,width=40, bg='#e8e6e6')
verificar_end.insert(END,"Escreva o endereço aqui")
canvas.create_window(140, 45, window=verificar_end)

verificarButton = Button(root_ends_routing,text="Verificar
endereço",width=15,command=lambda:aplicarVerificacao((osm_checkbox_var.get(), google_checkbox_var.get())),
bg='#579482', fg='white', font=('helvetica', 13, 'bold'))
canvas.create_window(355, 45, window=verificarButton)

verif_result_var = StringVar(root_ends_routing)
verif_result_var.set('')

# endereço inicial
end_inicial_label = Label(root_ends_routing, bg=cor_dos_labels,  text='Endereço inicial:',
font=('helvetica', 13,'bold'))
canvas.create_window(95, 150, window=end_inicial_label)

endereco_inicio = Entry(root_ends_routing,width=25, bg='#f5f0f0')
canvas.create_window(110, 180, window=endereco_inicio)

# endereço final
end_final_label = Label(root_ends_routing, bg=cor_dos_labels,  text='Endereço final:', font=('helvetica',
13,'bold'))
canvas.create_window(290, 150, window=end_final_label)

endereco_fim = Entry(root_ends_routing,width=25, bg='#f5f0f0')
canvas.create_window(310, 180, window=endereco_fim)

# horario de partida
#hr_partida_label = Label(root_ends_routing, bg=cor_dos_labels,  text='Data e horário de partida (00h -
23h59)', font=('helvetica', 13,'bold'))
#canvas.create_window(720, 150, window=hr_partida_label)
#horario_inicio = Entry(root_ends_routing,width=40, bg='#f5f0f0')
#horario_inicio.insert(END,str(datetime.strftime((datetime.now() + timedelta(days=1)), '%d/%m/%y %H:%M')))
#canvas.create_window(700, 180, window=horario_inicio)
```

```
# enderecos intermediarios
ends_rota_label = Label(root_ends_routing, bg=cor_dos_labels,  text='Endereços intermediários (um por
linha):', font=('helvetica', 13,'bold'))
canvas.create_window(195, 215, window=ends_rota_label)

#end_clientes = ScrolledText(root_ends_routing, xscrollcommand = scrollbar.set)
end_clientes = ScrolledText(root_ends_routing,  wrap="none")
scrollbar_endclientes = Scrollbar(root_ends_routing, orient=HORIZONTAL, command=end_clientes.xview)
#end_clientes['xscrollcommand'] = scrollbar.set
end_clientes.configure(xscrollcommand=scrollbar_endclientes.set)
scrollbar_endclientes.place(x=30,y=400,width=350)
end_clientes.place(x=30, y=235, width=350, height=170)

# Botão "Otimizar Rota"
canvas.create_window(460, 350, window=Label(root_ends_routing, text='→', font=('helvetica', 54))) # seta
otimizarButton = Button(root_ends_routing,text="Otimizar rota", height=2,
width=12,command=lambda:aplicarRotas(), bg='#229399', fg='white', font=('helvetica', 13, 'bold'))
canvas.create_window(460, 305, window=otimizarButton)


# enderecos otimizados
canvas.create_window(670, 215, window=Label(root_ends_routing, bg=cor_dos_labels, text='Resultado /
Endereços otimizados:', font=('helvetica', 13, 'bold')))
resultado_otimizado = ScrolledText(root_ends_routing, wrap="none")
scrollbar_resultado = Scrollbar(root_ends_routing, orient=HORIZONTAL, command=resultado_otimizado.xview)
resultado_otimizado.configure(xscrollcommand=scrollbar_resultado.set)
scrollbar_resultado.place(x=540,y=400,width=350)
resultado_otimizado.place(x=540, y=235, width=350, height=170)
resultado_otimizado.config(bg='#e6e6e6')

# Drop-down table
#servico_dropdown = OptionMenu(master, variable, "one", "two", "three")
servico_escolhido = StringVar(root_ends_routing)
servico_escolhido.set("OSRM")
canvas.create_window(460, 255, window=OptionMenu(root_ends_routing, servico_escolhido, "OSRM")) #
,"Google"))
servicoRotas_label = Label(root_ends_routing, bg='#d4d0cf',text=servico_escolhido.get(),
font=('helvetica', 9))
canvas.create_window(460, 390, window=servicoRotas_label)


root_ends_routing.mainloop()
```