UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Gabriel Dal Belo Gomes Santos

Avaliação de desempenho do algoritmo de Clarke-Wright para a construção de rotas e de seu aprimoramento utilizando busca local

Uberlândia, Brasil

2023

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Gabriel Dal Belo Gomes Santos

Avaliação de desempenho do algoritmo de Clarke-Wright para a construção de rotas e de seu aprimoramento utilizando busca local

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Sistemas de Informação.

Orientador: Dr. Paulo Henrique Ribeiro Gabriel

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Sistemas de Informação

Uberlândia, Brasil

2023

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Faculdade de Computação
Av. João Naves de Ávila, nº 2121, Bloco 1A - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4144 - http://www.portal.facom.ufu.br/ facom@ufu.br

ATA DE DEFESA - GRADUAÇÃO

Curso de Graduação em:	Bacharelado em Sistemas de Informação				
Defesa de:	FACOM31802 - Trabalho de Conclusão de Curso II				
Data:	21/06/2023	Hora de início:	10:00	Hora de encerramento:	11:22
Matrícula do Discente:	11811BSI221				
Nome do Discente:	Gabriel Dal Belo Gomes Santos				
Título do Trabalho:	Avaliação de desempenho do algoritmo de Clarke-Wright para a construção de rotas e de seu aprimoramento utilizando busca local				
A carga horária curricular foi cumprida integralmente?	(X) Sim ( ) Não				

Reuniu-se no Anfiteatro/Sala 1B132, Campus Santa Mônica, da Universidade Federal de Uberlândia, a Banca Examinadora, designada pelo Colegiado do Curso de Graduação em Sistemas de Informação, assim composta: Professores: Christiane Regina Soares Brasil - FACOM/UFU; Wendel Alexandre Xavier de Melo - FACOM/UFU; e Paulo Henrique Ribeiro Gabriel - FACOM/UFU), orientador do candidato.

Iniciando os trabalhos, o presidente da mesa, Dr. Paulo Henrique Ribeiro Gabriel, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao discente a palavra, para a exposição do seu trabalho. A duração da apresentação do discente e o tempo de arguição e resposta foram conforme as normas do curso.

A seguir o senhor presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir o candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

(X) Aprovado Nota 90.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Paulo Henrique Ribeiro Gabriel**, **Professor(a) do Magistério Superior**, em 21/06/2023, às 11:22, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Christiane Regina Soares Brasil**, **Professor(a) do Magistério Superior**, em 21/06/2023, às 11:23, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Wendel Alexandre Xavier de Melo**, **Professor(a) do Magistério Superior**, em 21/06/2023, às 11:25, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **4584441** e o código CRC **0D59C5A3**.

Dedico este trabalho, em primeiro lugar, à minha família, que sempre esteve ao meu lado e fez de tudo para me apoiar e motivar desde o início de minha vida acadêmica. Em segundo lugar, dedico a todos os corações apaixonados pelo conhecimento que anseiam por deixar sua marca no mundo e torná-lo cada vez melhor. Em terceiro lugar, dedico a todos aqueles que me admiram e sempre acreditaram no meu potencial, contribuindo de alguma forma para o florescer dele. Por fim, dedico a mim mesmo, como um lembrete de tudo aquilo que superei até aqui, e como uma visão de tudo aquilo que ainda posso conquistar.

Agradecimentos

Gostaria de agradecer aos amigos incríveis que conquistei durante essa jornada; amigos que estiveram ao meu lado me ajudando, sofrendo, se divertindo, chorando e passando pelas diversas emoções que essa montanha-russa nos proporcionou. Eu não teria chegado até aqui sem vocês, que fizeram dessa a melhor fase da minha vida. Obrigado Guilherme, Henrique, João Vitor, Railson, Vitor, Carlos e Davide. Fico feliz de saber que acompanharei o sucesso crescente que a vida lhes reserva, e espero que continuemos a ter nossas incríveis e divertidas reuniões enquanto a vida perdurar.

Também agradeço ao meu orientador, o professor Paulo, que foi tão solícito e compartilhou comigo as ideias mais interessantes que tinha referentes à disciplina que um dia cursei com ele, e que mesmo gostando bastante, eu nunca imaginei que viria a se tornar a base para este meu último e mais importante trabalho na faculdade. Obrigado pela oportunidade e por todo o suporte que me deu. O senhor foi um dos melhores professores que tive nessa jornada.

"O homem que nunca muda de opinião é como água parada, e alimenta répteis da mente" - William Blake

Resumo

O Problema de Roteamento de Veículos (PRV) representa, ainda nos dias atuais, um grande desafio devido à sua complexidade computacional. Não há um método eficiente capaz de resolver todos os possíveis casos, que variam desde pequenas instâncias até instâncias enormes contendo dezenas de milhares de pontos a serem considerados. Neste trabalho, retrocedemos um pouco no tempo ao analisar e comparar em termos de eficiência e desempenho duas das primeiras soluções mais abrangentes para este problema, sendo a primeira o algoritmo de Clarke e Wright, proposto em 1964, e a aplicação do método 2-opt, proposto alguns anos antes em 1958, em cima do resultado gerado pelo algoritmo anterior. A aplicação desse método de melhoria em cima de uma heurística construtiva aprimora o resultado ainda mais e o aproxima da solução ótima a um baixo custo computacional. Ambos os algoritmos foram implementados manualmente utilizando a linguagem de programação Python, permitindo execuções nas quais podemos observar de maneira direta o custo das soluções resultantes.

Palavras-chave: Roteamento de veículos. Heurísticas. Algoritmo de Clarke-Wright. Busca local.

Abstract

The Vehicle Routing Problem (VRP) still represents a significant challenge in modern days due to its computational complexity. There is no efficient method capable of solving all possible cases, which range from small instances to massive instances containing tens of thousands of points to be considered. In this work, we take a step back in time to analyze and compare two of the earliest comprehensive solutions to this problem in terms of efficiency and performance. The first one is the Clarke and Wright algorithm proposed in 1964, and the second is the application of the 2-opt method, proposed a few years earlier in 1958, on the result generated by the previous algorithm. Applying this improvement method on top of a constructive heuristic further enhances the result. It brings it closer to the optimal solution at a low computational cost. Both algorithms were manually implemented using the Python programming language, enabling direct executions where we can directly observe the cost of the resulting solutions.

Keywords: Vehicle routing. Heuristics. Clarke-Wright algorithm. Local search.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

TSP	*Travelling Salesman Problem* (Problema do Caixeiro Viajante).
VRP	*Vehicle Routing Problem* (Problema do Roteamento de Veículos).
NP	*Non-deterministic Polynomial time* (Polinomial Não Determinístico).

Sumário

1 Introdução

Mesmo em tempos modernos, problemas de escalonamento de rotas, como o do Caixeiro Viajante e do Roteamento de Veículos, ainda representam desafios significativos na área da computação. Esses problemas podem variar desde casos básicos, com um número relativamente pequeno de variáveis, até casos mais complexos que se aproximam da realidade, nos quais encontramos números muito mais elevados de elementos a serem considerados.

A literatura apresenta experimentos grandiosos realizados nesse campo. No entanto, o maior número de pontos que pode ser considerado em uma resolução computacional com velocidade aceitável ainda é extremamente baixo quando comparado às possíveis aplicações reais desses problemas, onde podemos encontrar quantidades surpreendentemente altas de nós a serem analisados.

1.1 Objetivo

O objetivo deste trabalho é realizar uma comparação e análise de desempenho entre duas diferentes aplicações do método de economias de Clarke e Wright (1964) para a resolução de casos específicos do Problema do Roteamento de Veículos. Será enfatizada a melhoria alcançada na transição da primeira versão para a segunda, incorporando o uso da busca local 2-opt (CROES, 1958).

A pesquisa foi desenvolvida em cinco etapas: revisão bibliográfica, estudo dos algoritmos implementados, implementação e execução dos algoritmos, análise e comparação dos resultados, e registro das conclusões. O objetivo é obter uma visão clara e simplificada da melhoria alcançada pela transição do método original para sua versão aprimorada, além de um entendimento pleno do funcionamento de ambos os métodos.

1.2 Justificativa

O Problema do Roteamento de Veículos pode ser descrito como a execução em paralelo de múltiplas instâncias do Problema do Caixeiro-Viajante. Neste trabalho, buscamos demonstrar a análise comparativa proposta, evidenciando essa relação entre os problemas e destacando que um é composto por várias instâncias diferentes, porém complementares, do outro.

Embora este estudo não apresente resultados inéditos, considerando os estudos mais aprofundados e complexos realizados nas últimas décadas, essa análise é válida por

se tratar de uma revisão simples, direta e atualizada, com foco específico em um único método. Além disso, serão apresentadas comparações de eficiência entre os dois algoritmos aplicados em sequência.

1.3 Metodologia

A metodologia adotada neste estudo envolveu a implementação, teste e comparação de dois algoritmos diferentes para a resolução de casos específicos do Problema do Roteamento de Veículos. O primeiro algoritmo corresponde à implementação original do método de economias de Clarke e Wright, enquanto o segundo algoritmo representa uma versão aprimorada, que incorpora a busca local 2-opt.

Ambos os algoritmos foram implementados utilizando a linguagem de programação Python. Os dados de entrada para os algoritmos foram fornecidos por meio de arquivos de texto no formato CSV, que continham informações sobre a quantidade de carga a ser coletada em cada ponto da rota e a distância entre cada par de pontos.

A partir dessas implementações, os algoritmos foram executados e seus desempenhos foram analisados e comparados. O objetivo principal foi avaliar a diferença de desempenho entre os algoritmos, especialmente em relação ao aumento da complexidade dos casos estudados.

2 Referencial Teórico

Neste capítulo, são apresentados os conceitos fundamentais e os trabalhos correlatos para fornecer um embasamento teórico sobre os problemas tratados neste.

Primeiramente, seremos apresentados à definição do problema. Na sequência, conheceremos superficialmente algumas das várias versões existentes - incluindo aquela que será considerada neste estudo - e as classificações dos principais métodos de solução.

Em segundo lugar, conheceremos de maneira detalhada os dois algoritmos que serão abordados neste trabalho, e como atuam na solução do problema.

Ao final do capítulo, revisaremos os principais trabalhos que deram embasamento e contribuíram para o avanço dos estudos referentes à este problema.

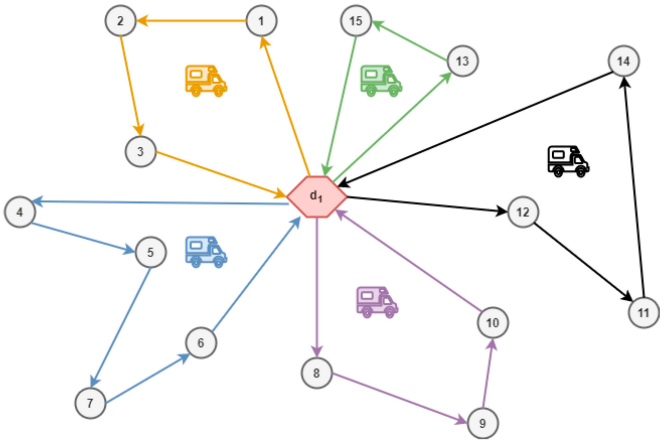
2.1 O Problema do Roteamento de Veículos

O Problema do Caixeiro Viajante (PCV), conhecido em inglês como *Travelling Salesman Problem* (TSP), é caracterizado pela necessidade de um mercador fictício visitar n cidades em sua rota de viagem, percorrendo cada cidade uma única vez, enquanto busca pelo caminho de menor custo possível e retornando à cidade de partida como último ponto da rota. As primeiras formulações do problema datam desde o início do século XIX, mas a formulação matemática propriamente dita surgiu apenas na década de 1930 na forma de um problema de roteamento de ônibus escolares (LAWLER et al.).

Por sua vez, o Problema do Roteamento de Veículos (PRV), também conhecido como *Vehicle Routing Problem* (VRP) em inglês, surgiu na década de 1950 em uma situação prática onde era necessário organizar a distribuição de gasolina para diversas estações de venda por meio de caminhões (DANTZIG; RAMSER, 1959).

Desde sua formulação inicial, o PRV foi descrito como uma generalização do Problema do Caixeiro Viajante (PCV), pois apresenta uma situação semelhante onde vários veículos fictícios partem de um mesmo ponto central, percorrem rotas específicas onde visitam até vários clientes uma única vez, e retornam ao ponto inicial — tudo isso também priorizando uma rota de menor custo. Uma exemplificação do PRV pode ser vista na Figura 1.

Figura 1 – Exemplificação do PRV. Neste exemplo, vemos 15 pontos de entrega a serem atendidos e um único depósito central, com as rotas já definidas após uma execução fictícia do algoritmo. Ao fim da execução, temos cinco rotas e, consequentemente, cinco veículos atendendo-as.



Fonte: Adaptado de Kovács, Agárdi e Bányai (2020).

Ambos os problemas estão entre os mais difíceis de otimização combinatória, que define problemas sob um domínio finito composto por valores discretos. Devido à sua complexidade, que cresce em uma velocidade maior do que exponencial, métodos e algoritmos exatos rapidamente perdem sua eficiência. Por exemplo, uma instância do PCV com $n = 15$ apresenta mais de 600 trilhões[1] de rotas possíveis (DANTZIG; RAMSER, 1959).

Tais características colocam ambos os problemas na classe $\mathcal{NP}$-difícil. Ainda não há um algoritmo eficiente que possa resolvê-los em todos os casos e não se sabe se um dia será encontrado, no entanto, existem algoritmos de aproximação que podem encontrar soluções razoáveis em tempo hábil para muitos casos (GRECO, 2008).

2.1.1 Versões do PRV

Diversas versões do problema foram apresentadas e exploradas na literatura desde sua formulação inicial. Entetanto, é importante destacar que um mesmo caso pode ser caracterizado como duas ou mais versões do PRV. Algumas dessas versões são:

- PRV Capacitado (no inglês, CVRP), onde consideramos um único depósito central de onde todos os veículos partem, e para onde retornam, além de uma capacidade uniforme dos veículos.
- PRV com janelas de tempo (no inglês, VRPTWs), que consideram clientes disponíveis apenas durante um período específico de tempo.

[1] O valor exato é 653.837.184.000.000

- PRV com retiradas e entregas, onde cada veículo busca e entrega produtos em diversos locais.
- PRV com transporte de volta (do inglês, VRPB), que inclui um conjunto de clientes para os quais os produtos devem ser entregues e um conjunto de fornecedores cujas mercadorias precisam ser transportadas de volta ao centro de distribuição.
- PRV com entrega dividida (do inglês, SDVRP), onde cada cliente pode ser servido por mais de um veículo.
- PRV Periódico (do inglês, PVRP), onde o período planejado para realização das entregas considera dias, diferente do padrão que as considera como planejadas para um único dia.
- PRV com frota heterogênea (do inglês, HFVRP), onde cada veículo possui diferentes capacidades e custos.
- *Dial-a-ride problem* (DARP), que envolve um conjunto de rotas de custo mínimo para satisfazer determinadas solicitações de transporte. Ao invés de produtos, temos usuários partindo de um conjunto de origens (pontos de coleta) para um conjunto de destinos (pontos de entrega). Por se tratarem de pessoas sendo transportadas, os critérios e especificações se tornam mais rígidos e complexos.
- PRV Estocástico (do inglês, SVRP), que considera valores aleatórios para aspectos e probabilidades especificas, como a demanda de um determinado cliente e a probabilidade do veículo estar percorrendo determinada rota.
- PRV Dinâmico (do inglês, DVRP), que considera a probabilidade de aparição dos clientes, tempos de viagem, tempos de serviço, e até mesmo disponibilidade de veículos.
- Problema do Roteamento de Inventário (do inglês, IRP), que integra gerenciamento de estoque e decisões de agendamento ao PRV padrão.

2.1.2 Métodos de Solução

Até o momento, não existem abordagens eficientes em termos de tempo de processamento e memória que forneçam uma solução exata geral para o PCV e o PRV. Por outro lado, existem métodos heurísticos que se provaram bastante eficientes em casos específicos de grande porte, sendo utilizados para encontrar uma solução inicial viável para que então sejam aplicados algoritmos exatos para convergir à solução ótima.

Para a resolução tanto do PCV quanto do PRV, os métodos heurísticos são classificados em dois tipos: **construtivos** e **de melhoria**. Os algoritmos construtivos partem de

um conjunto vazio e constroem rotas iterativamente até que todos os clientes sejam cobertos por alguma rota, como o algoritmo de Clarke e Wright (1964), citado anteriormente, e o método de Mole e Jameson (1976) que aprimorou a heurística de Clarke e Wright. Os trabalhos de melhoria de heurística com múltiplas rotas de Christofides, Mingozzi e Toth (1981), Thompson e Psaraftis (1993) e Kindervater e Savelsbergh (2003) também compõem este tipo. Esses métodos podem ser sequenciais, expandindo uma rota de cada vez, ou paralelos, desenvolvendo duas ou mais rotas simultaneamente.

Os algoritmos de melhoria, por outro lado, partem de uma solução já existente e buscam melhorá-la o máximo possível. Esses métodos podem operar em uma única rota por vez (IRSR, do inglês *intra-rotas-single-route*) ou múltiplas rotas (IRMR, do inglês *inter-rotas-multi-route*), sendo construídos em cima de uma heurística k-opt, onde uma vizinhança é formada e alterada através de testes cíclicos nos quais ocorre a substituição de k arestas por outras k arestas que não estão nesse ciclo. Um dos principais e mais básicos métodos de melhoria é o 2-opt, algoritmo de busca local proposto por Croes (1958).

O PRV em específico possui ainda um método heurístico exclusivo: o de **duas fases**, aplicado nos trabalhos de Fisher e Jaikumar (1981)) e nos algoritmos pétala (RYAN; HJORRING; GLOVER, 1993) e de varredura (GILLETT; MILLER, 1974). Nesse método, a solução do problema é dividida nas fases de Agrupamento e Roteamento. Na primeira fase, são definidos grupos de clientes, sendo que cada grupo representa uma rota. Já na segunda fase, é estabelecida a rota em si. A ordem na qual as duas fases são executadas pode ser alternada, com diferentes algoritmos associados a cada ordem (WILHELM, 2015).

2.2 Algoritmos

Entre os dois algoritmos aqui explorados, o primeiro consiste na implementação original da heurística de Clarke-Wright, aplicada considerando um PRV Capacitado. O segundo consiste na aplicação da heurística 2-opt para melhoria do resultado alcançado pelo primeiro algoritmo.

2.2.1 Heurística de Clarke-Wright

Considere um único depósito central D e n pontos a serem atendidos por ele. A ideia inicial consiste em despachar n veículos de mesma capacidade de transporte, um para cada ponto a ser atendido, somando os custos individuais, que são definidos apenas como a distância entre dois pontos, para se obter o custo total da viagem.

$$2\sum_{i=1}^{n} d(D,i)$$

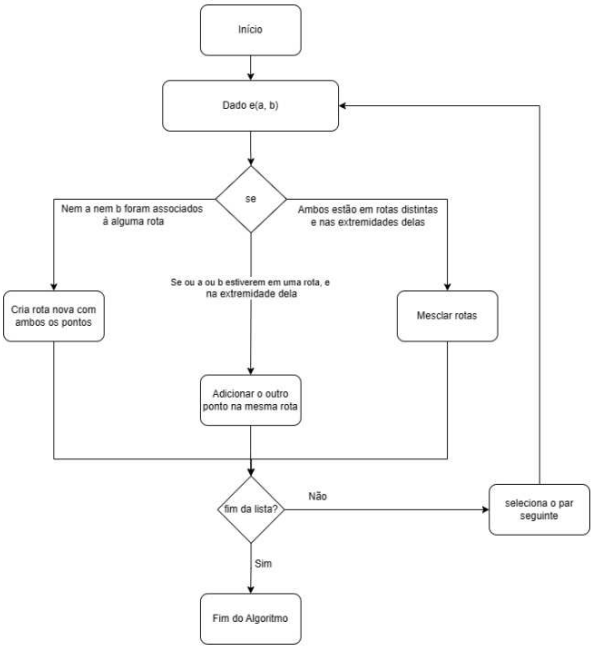
Se for definido que cada veículo servirá dois pontos distintos, a e b, ao invés de um único, temos que o valor de economia e para o trajeto composto por a e b será dado pela soma da distância do depósito para ambos os pontos individualmente subtraída pela distância entre ambos os pontos.

$$e(a, b) = d(D, a) + d(D, b) - d(a, b)$$

Quanto maior o valor resultante mais desejável se mostra a adição deste trajeto à uma rota, pois um valor alto mostra o quão vantajosa é a combinação destes dois pontos em um único trajeto, a menos que o mesmo acabe violando uma ou mais restrições do problema. Considerando isto, temos o seguinte algoritmo em sua versão reformulada por Larson e Odoni (1981):

1. Calcule o valor $e(a, b)$ para cada par (a, b) de pontos a serem visitados.
2. Ranqueie os valores em ordem decrescente para formar uma lista.
3. Inclua um par (a, b) em uma rota se nenhuma restrição (como a quantidade transportada por cada veículo, possíveis restrições de distância total percorrida ou números de pontos visitados, entre outras) for violada, e se uma das três condições a seguir for verdadeira:
   a) Nem a nem b foram associados a alguma rota. Uma rota então é inicializada com ambos.
   b) Se apenas um ponto dentre a e b já faz parte de uma rota e este se encontra em uma das extremidades dela desconsiderando o depósito D. Nesse caso, adicionamos o outro ponto antes ou após àquele que já pertence à rota, tornando ele o novo adjacente ao depósito.
   c) Se ambos os pontos fazem parte de rotas distintas e estão em extremidades delas. Nesse caso, ambas as rotas são combinadas.
4. Enquanto a lista não for totalmente explorada ou todos os pontos tiverem sido visitados, repita o passo anterior e introduza a próxima entrada. Quando uma das duas condições for alcançada, pare.

Figura 2 – Fluxograma do algoritmo de CW

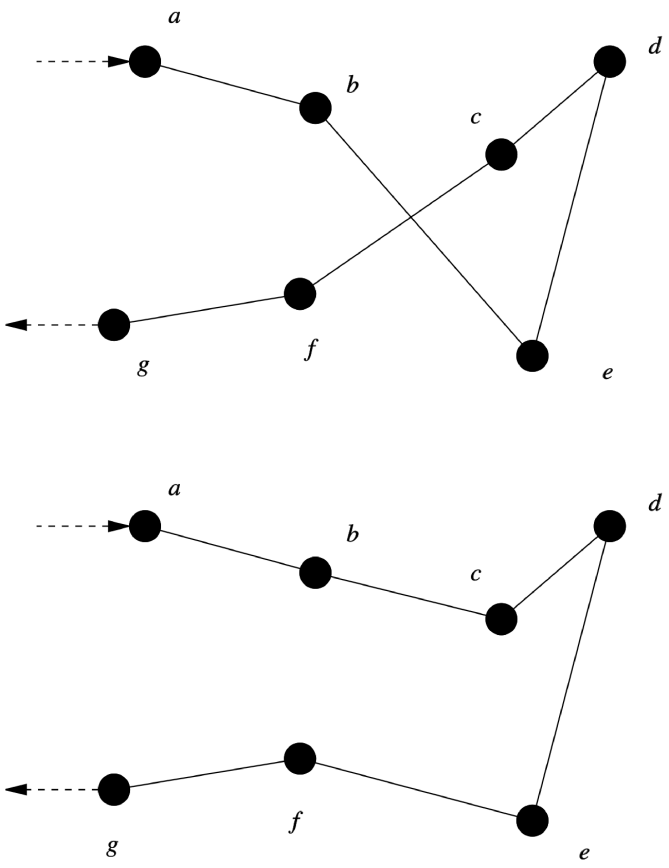


Fonte: Autoria própria.

2.2.2 Busca Local 2-opt

Partindo da solução fornecida pelo algoritmo anterior, aplicamos o algoritmo da busca local 2-opt (CROES, 1958), que representa uma das mais famosas heurísticas de melhoria. Esse algoritmo é fundamentado na troca de posição entre dois pontos i e j de uma mesma rota, visando otimizá-la através de uma possível redução de custos derivada dessa troca e cortar possíveis cruzamentos. A Figura 3 exemplifica a execução desse algoritmo.

Figura 3 – Aplicação do 2-opt. No exemplo, vemos uma rota formada por oito pontos, havendo um cruzamento dos pares (b, e) e (c, f). Com a aplicação da busca local 2-opt, o cruzamento é desfeito e ocorre uma possível redução de custos com o trajeto total.



Fonte: Adaptado de Pierreselim (2009).

Todos os possíveis pares de pontos dentro da rota são considerados, exceto o depósito. Caso a troca resulte em uma nova versão da rota cujo custo é menor do que o da rota original, ocorre a substituição da mesma. A troca dos pares ocorre seguindo os seguintes passos:

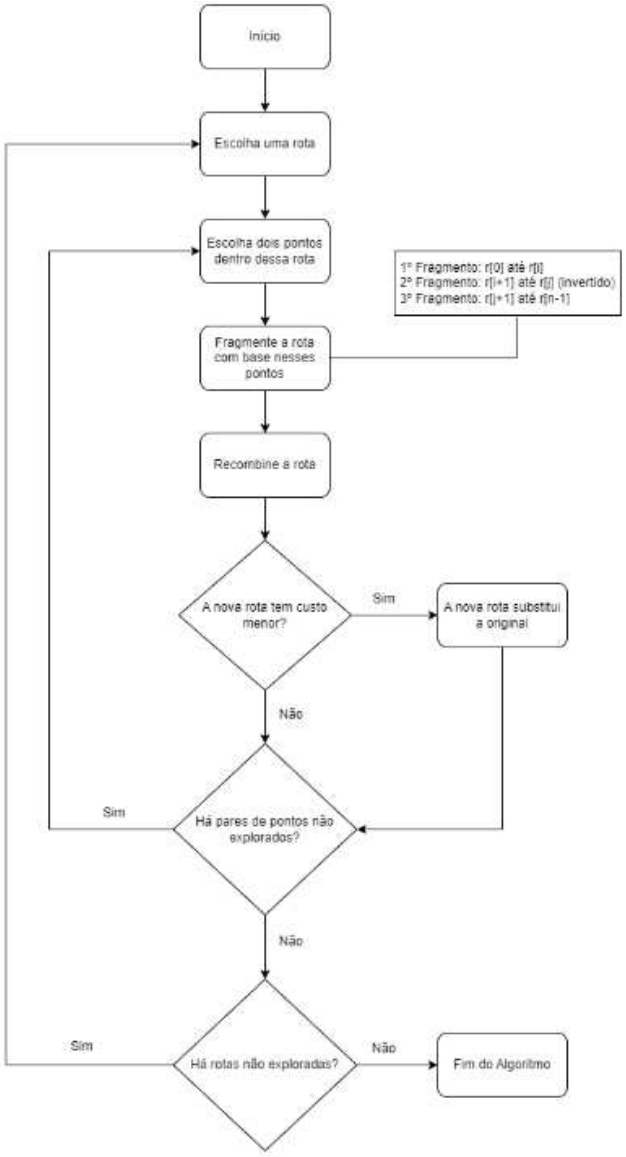
1. Escolhe-se dois pares de elementos dentro de uma mesma rota.
2. Essa rota é fragmentada em três partes:
   a) A primeira vai do primeiro elemento da rota (índice 0) até o primeiro elemento da troca (índice i).
   b) A segunda vai do elemento posterior ao primeiro da troca (índice $i + 1$) até o segundo elemento da troca (índice j).
   c) A terceira do elemento posterior ao segundo da troca (índice $j+1$) até o último elemento da rota (índice $n - 1$).

3. A rota alternativa é criada combinando as três partes da rota original em ordem, mas com a segunda parte sendo invertida antes da combinação.

4. Se o custo da rota alternativa se mostra menor que o custo da rota original, a rota alternativa a substitui. Se não, a rota original se mantém.

5. Se ainda há pares de elementos da rota que não forem escolhidos, parte-se para o próximo e retorna-se ao segundo passo.

Figura 4 – Fluxograma do algoritmo 2OPT



Fonte: Autoria própria.

Após a execução do algoritmo em cada uma das rotas, tem-se uma versão mais próxima de ótima de todas elas.

2.3 Trabalhos Correlatos

Entre os estudos de diversos autores que foram considerados neste trabalho, destacam-se algumas pesquisas que contribuíram significativamente para a melhoria das resoluções do problema tratado. Dantzig e Ramser (1959), em seu trabalho sobre o Problema do Despacho de Veículos, apresentaram a formulação do PRV pela primeira vez, exemplificando a complexidade maior do que exponencial do problema.

Um ano antes, Croes (1958) propôs a busca local 2-opt, uma das primeiras e mais simples soluções propostas para solucionar o PCV.

Clarke e Wright (1964) desenvolveram a primeira e mais famosa heurística para o PRV, baseada no conceito de economias (*savings*). Essa heurística ficou conhecida como Clarke-Wright e se tornou uma referência nessa área (mais detalhes sobre essa heurística serão apresentados no Capítulo 3). Posteriormente, Mole e Jameson (1976) apresentaram um método baseado na geração sequencial de rotas de veículos que aprimorou a heurística de Clarke-Wright. Esse método demonstrou ser computacionalmente eficiente ao resolver um problema de depósito com 225 clientes, conforme descrito em seu trabalho.

Outra técnica de melhoria é a inversão de rotas, aplicada por Bennett e Gazis (1972) em uma adaptação do PRV em um contexto de transporte escolar. Tal técnica é aplicada durante o processamento e não só proporcionou uma redução de custos, como também criou a possibilidade de se combinar rotas que não poderiam ser combinadas normalmente.

Gillett e Miller (1974) descreveram o Algoritmo de Varredura, um método heurístico de duas fases. Este algoritmo foi capaz de encontrar uma solução viável para um problema com 100 clientes em aproximadamente dois minutos. Golden, Magnanti e Nguyen (1977) propuseram modificações e extensões nos algoritmos formulados pelos autores anteriores, permitindo resolver problemas com centenas de clientes em questão de segundos. Eles também apresentaram um algoritmo altamente eficiente para roteamento envolvendo múltiplos depósitos.

Christofides, Mingozzi e Toth (1981) aplicaram algoritmos de busca em árvore com melhorias de redução do problema, bem como testes de dominância. Fisher e Jaikumar (1981) detalharam uma heurística que se mostrou altamente eficiente na época, com duas fases e capaz de encontrar pelo menos uma solução viável, caso exista. Além disso, consideraran a capacidade de cada veículo ao definir as rotas.

Thompson e Psaraftis (1993) investigaram a aplicação de uma nova classe de algoritmos de busca de vizinhança, conhecidos como algoritmos de transferências cíclicas, para roteamento multi-veícular e resolução de problemas de escalonamento. Esse trabalho é considerado o primeiro algoritmo de melhoria multi-rota.

Ryan, Hjorring e Glover (1993) demonstraram ideias para o aprimoramento do método pétala, desenvolvido previamente por Gillett e Miller (1974) por meio do Algoritmo de Varredura. O método pétala continua sendo um dos mais eficientes e amplamente utilizados até os dias atuais. As melhorias propostas por Ryan, Hjorring e Glover (1993) permitem o estabelecimento de rotas que não seguem o padrão de pétala, embora ainda se baseiem nele.

Van Breedam (1995) relatou o uso de métodos de melhoria baseados em recozimento simulado, uma técnica de busca local probabilística. Por fim, Kindervater e Savelsbergh (2003) exploraram o método de troca de vizinhos e demonstrou melhorias no contexto de múltiplas rotas em seu trabalho.

Em Pichpibul e Kawtummachai (2012) foi proposto um algoritmo que foi aprimorado a partir do algoritmo clássico de economias de Clarke e Wright para resolver o PRV Capacitado. A ideia principal do algoritmo proposto foi a de combinar o algoritmo original com seleções no modelo torneio e roleta para determinar um algoritmo novo e eficiente.

Pillac et al. (2013) apresentaram uma descrição geral do roteamento dinâmico, introduzindo a noção de grau de dinamismo e apresentando uma revisão abrangente de aplicações e métodos de solução para problemas de roteamento dinâmico de veículos.

Montoya-Torres et al. (2015) apresentaram uma nova revisão que considerou artigos publicados entre 1988 e 2014, nos quais diversas variantes do modelo multi-depósitos do PRV são estudadas: com janelas de tempo, entrega fracionada, frota heterogênea, entregas periódicas, e coleta e entrega. A revisão também classifica as abordagens de acordo com os objetivos únicos ou múltiplos que são otimizados. Algumas linhas para pesquisas futuras também foram apresentadas.

Esses estudos apresentaram avanços significativos no campo do roteamento de veículos, contribuindo para a melhoria das resoluções do problema ao longo dos anos. Desde a formulação inicial do PRV até as heurísticas e algoritmos mais recentes, essas pesquisas têm impulsionado a eficiência e a capacidade de lidar com problemas cada vez maiores e mais complexos no campo do roteamento de veículos.

3 Desenvolvimento

Neste capítulo, a implementação de cada algoritmo será detalhada e os resultados da execução de ambos serão demonstrados e analisados para fins de comparação.

3.1 Primeiro Algoritmo

O algoritmo de Clarke-Wright foi implementado na linguagem de programação Python, conforme mostrado na Figura 9, e possui quatro funções principais.

1. *initializePoints* recebe a quantidade total de pontos e inicializa um dicionário que registrará quando cada um for visitado e consequentemente se tornar parte de alguma rota, além de também registrar a rota em si ao qual ele estará vinculado.

2. *loadQuantities* realiza a leitura do arquivo que contém a listagem dos pontos e suas respectivas quantidades de carga demandadas. Os valores são armazenados em um dicionário, e suas respectivas chaves são os próprios pontos, listados numericamente.

3. *loadDistances* realiza a leitura do arquivo que contém a listagem de cada par de pontos e a distância/custo da viagem entre eles. Tudo isso é armazenado em uma lista no formato de arranjo, e cada valor de distância é acessado com seu respectivo par de pontos sendo passado como índice dessa lista.

4. *calculateSavings* recebe a lista gerada pela função anterior e calcula os valores de economia, que serão considerados para a construção das rotas. Esses valores são devolvidos ordenados em ordem decrescente, no formato de uma lista semelhante a que é passada como parâmetro desta função.

Figura 5 – Código principal

```
import sys
import time
from datetime import datetime
from cws _functions_v3 import

vehicleCapacity = int(sys.argv[1])
nodes = int(sys.argv [2])       #This count will always include the depot.
file_q = str(sys.argv [3])      #The path to the file with the quantity values.
file_d = str (sys.argv [4])     #The path to the file with the distance values.

visitedPoints = initializePoints (nodes)
#Will register if each node has been visited.
quantityPerNode = loadQuantities (file_q)
#Will storage the quantity value associated with each node.
linksList = loadDistances (file_d, nodes)
#Will storage the distance values between each pair of nodes.
sortedSavings = calculateSavings(linksList, nodes)
#Will storage the sorted savings values.

start time = datetime.now()
routes = executewSalgorithm(sortedSavings, quantityPerNode, vehicleCapacity, visitedPoints)
routes = calculateCost(linksList, routes)
time.sleep (1)
end time = datetime.now()
routeCounter = 1
for route in routes:
    print ("Route + str(routeCounter) + ":" + str(route.path))
    print ("Quantity of Products: " + str(route quantity))
    print("Total Cost: " + str(route.cost))
    print ("")
    routeCounter += 1
print("Execution Time: " + str(end time - start time))
```

Fonte: Autoria própria.

Após a execução de todas essas funções, seus resultados são passados como parâmetro para a função *executeCWSalgorithm*, que como o nome sugere, realiza a execução do algoritmo e a construção das rotas. Seu resultado é passado como parâmetro da função *calculateCost*, que percorre as rotas já construídas para calcular o custo de cada uma.

O programa foi executado de maneira direta via linha de comando (Figura 6) em uma pasta contendo todos os arquivos necessários, com sua saída sendo direcionada para um arquivo de texto. O programa é executado utilizando os seguintes parâmetros: *i*) capacidade de carga para os veículos; *ii*) número total de pontos a serem considerados; *iii*) nome do arquivo contendo os pares ponto e quantidade de carga a ser entregue neste ponto; e *iv*) nome do arquivo contendo a listagem da distância entre todos os pares de pontos. Esses dois arquivos estão no formato *csv*.

Figura 6 – Execução via linha de comando

```
python cws_with_functions.py 23 10 node_quantities.csv node_links.csv > output.txt
```

Fonte: Autoria própria.

Foram avaliados cinco exemplos. No primeiro exemplo, extraído de Larson e Odoni (1981), a capacidade de carga de cada veículo é 23 e há nove pontos a serem visitados que, contando com o depósito, totalizam $n = 10$. A Figura 7 mostra os valores de custo entre cada par de pontos neste caso onde $d(a, b) = d(b, a)$. Após a execução, as rotas foram definidas como mostrado na Tabela 1.

Figura 7 – Valores de custo do Exemplo 1

	1	2	3	4	5	6	7	8	9	10
1										
2	25									
3	43	29								
4	57	34	52							
5	43	43	72	45						
6	61	68	96	71	27					
7	29	49	72	71	36	40				
8	41	66	81	95	65	66	31			
9	48	72	89	99	65	62	31	11		
10	71	91	114	108	65	46	43	46	36	

Fonte: Larson e Odoni (1981).

Tabela 1 – Resultado do 1º exemplo

Pontos p/Rota	Quantidade de Produtos	Custo Total
5	23	203
4	19	194

Já no segundo exemplo, retirado de Chegg (2018), a capacidade de carga de cada veículo é 100 e há oito pontos a serem visitados que, contando com o depósito, totalizam $n = 9$. A Figura 8 mostra os custos entre cada par de pontos, seguindo o mesmo padrão do exemplo anterior. Na Tabela 2, é mostrado o resultado dessa execução.

Figura 8 – Valores de custo do Exemplo 2

	1	2	3	4	5	6	7	8	9
1	-								
2	26	-							
3	15	15	-						
4	20	23	24	-					
5	7	26	13	26	-				
6	25	33	20	42	18	-			
7	16	40	27	32	14	25	-		
8	24	38	35	15	31	49	32	-	
9	29	54	43	39	32	45	20	30	-

Fonte: Chegg (2018).

Tabela 2 – Resultado do 2º exemplo

Pontos p/Rota	Quantidade de Produtos	Custo Total
2	40	59
4	95	86
2	53	65

Por envolverem um domínio relativamente pequeno de pontos, ambas as execuções tiveram um resultado gerado em tempo quase imperceptível mesmo quando multiplicado por 1000, sendo registrado como 0.0 mili segundo.

Para o terceiro exemplo, um caso maior foi gerado e testado especificamente para este experimento. Foram considerados 150 pontos, com valores de distância variando de 11 a 114 (respectivamente o menor e o maior valor de distância entre os pares de pontos do primeiro exemplo) e veículos com capacidade de carga igual a 1000. Foram criadas cinco rotas e o tempo de execução registrado foi de 114.312649 mili segundos, salto notável se comparado aos dois exemplos anteriores.

Tabela 3 – Resultado do 3º exemplo

Pontos p/ Rota	Quantidade de Produtos	Custo
41	993	984
41	1000	875
39	988	953
26	657	656
2	56	86

A existência da quinta rota mostra a possibilidade de criação de “rotas excedentes” que aparentemente não se mostram muito vantajosas, pois a quantidade de produtos

entregues e/ou pontos atendidos é tão pequena que a mesma poderia ser mesclada a uma das rotas anteriores. Ainda que o terceiro passo do algoritmo execute a combinação de rotas menores para evitar este cenário, o mesmo pode continuar ocorrendo devido a inviabilidade de se percorrer o caminho entre dois pontos específicos, possibilidade que existe diante de um valor de economias baixo demais ou até mesmo inexistente, ou por este ter sido analisado previamente mas a conexão ter sido rejeitada por alguma restrição ser violada durante a execução do algoritmo.

Abstraindo para um cenário real, imagine um bairro que possui a demanda de três entregas e outro no extremo oposto da mesma cidade que possui uma demanda de 20 entregas. Ainda que o veículo destinado ao segundo bairro possua espaço para os produtos da primeira entrega, a grande distância entre os bairros torna mais viável a operação de dois veículos do que um só para cobrir ambas as rotas.

No quarto exemplo, extraído de Fischetti, Toth e Vigo (1994), foram considerados 71 pontos e capacidade de carga igual a 1000. O tempo de execução registrado foi de 15.70 milisegundos.

Tabela 4 – Resultado do 4º exemplo

Pontos p/Rota	Quantidade de Produtos	Custo
31	991	1912
27	992	1568
12	424	729

Finalmente no quinto exemplo, também extraído de Fischetti, Toth e Vigo (1994), foram considerados 34 pontos e capacidade de carga igual a 1000. O resultado apresentou a mesma condição destacada no terceiro exemplo: a existência de uma rota excedente (a terceira) que, embora pudesse supostamente ser combinada à primeira, resultaria em uma grande rota inviável. O tempo de execução foi de 18.665 mili segundos.

Tabela 5 – Resultado do 5º exemplo

Pontos p/Rota	Quantidade de Produtos	Custo
14	757	727
16	927	974
3	216	76

3.2 Segundo Algoritmo

Uma cópia do arquivo do primeiro algoritmo foi feita e, nela, novas funções referentes ao segundo algoritmo foram implementadas:

1. *calculateCostII* é uma versão modificada da função *calculateCost*, implementada junto com o primeiro algoritmo. Enquanto a versão original percorria uma lista com todas as rotas e efetuava o cálculo de custo total em cada uma delas, esta versão é mais simples e aplicada manualmente em uma única rota. Essa nova versão se mostrou necessária diante da simplicidade do segundo algoritmo.

2. *TWOPTswap* cria a rota alternativa com base na rota original passada como parâmetro. Em suma, executa o segundo e terceiro passo do 2-opt.

3. *execute2OPTalgorithm* utiliza das duas funções anteriores para execução dos demais passos do algoritmo, percorrendo a rota passada como parâmetro enquanto testa todas as trocas de elementos e verifica os novos custos. É utilizada dentro de um laço que percorre a lista de rotas, executando o algoritmo em uma rota por iteração.

Figura 9 – Código do algoritmo 2-opt

```
def TWOPTswap(path, i, j):
    #Removes the depot (1) from the begin and the end of the route.
    path.pop(0)
    path.pop(len(path)-1)
    #Reverses the second subroute.
    mid_subroute = path[i+1:j+1]
    mid_subroute.reverse()
    #Creates the new route and add the depot again.
    path = [1] + path[0:i+1] + mid_subroute + path[j+1::] + [1]
    return path

def execute2OPTalgorithm(linksList, route):
    #The best route initially is the original route.
    best = route
    foundImprovment = True
    while (foundImprovment):
        foundImprovment = False
        for i in range(len(route.path)-2):
            for j in range(i+1, len(route.path)-2):
                    #Creates a copy of the original route.
                    newPath = route.path[:]
                    #Storages the alternate route.
                    newPath = TWOPTswap(newPath, i, j)
                    newCost = calculateCostII(linksList, newPath)
                    if newCost < best.cost:
                        #Updates the new best route.
                        best.path = newPath
                        best.cost = newCost
                        route = best
                        foundImprovment = True
    return best
```

Fonte: Autoria própria.

Figura 10 – Trecho alterado do programa original

```
start_time = datetime.now()
routes = executeCWSalgorithm(sortedSavings, quantityPerNode, vehicleCapacity, visitedPoints)
routes = calculateCost(linksList, routes)
for route in routes:
      route = execute2OPTalgorithm(linksList, route)
time.sleep(1)
end_time = datetime.now()
```

Fonte: Autoria própria.

Os mesmos exemplos utilizados no primeiro algoritmo foram reutilizados neste, com novos arquivos de saída sendo gerados. Comparando cada arquivo de saída do segundo algoritmo com sua contraparte do primeiro, é possível verificar a alteração sofrida em cada rota, assim como sua respectiva redução de custo. A execução deste algoritmo também se deu via linha de comando, seguindo o mesmo padrão do primeiro algoritmo.

Figura 11 – Execução via linha de comando

```
python cws_2opt.py 23 10 node_quantities.csv node_links.csv > output1b.txt
```

Fonte: Autoria própria.

Nos dois primeiros exemplos, nenhuma melhoria de rota foi alcançada, ou seja, todas as rotas envolvidas e seus respectivos custos permaneceram sem alteração. O tempo de execução do primeiro exemplo aqui já se tornou notável - 1.085 mili segundos, enquanto no segundo exemplo permaneceu 0.0 mili segundo.

Tabela 6 – Resultado do 1º exemplo

Pontos p/Rota	Quantidade de Produtos	Custo Inicial	Custo Final
5	23	203	203
4	19	194	194

Tabela 7 – Resultado do 2º Exemplo

Pontos p/Rota	Quantidade de Produtos	Custo Inicial	Custo Final
2	40	59	59
4	95	86	86
2	53	65	65

Do terceiro exemplo em diante, houveram rotas que foram alteradas e tiveram seus custos nitidamente reduzidos.

Tabela 8 – Resultado do 3º Exemplo

Pontos p/Rota	Quantidade de Produtos	Custo Inicial	Custo Final
41	993	984	821
41	1000	875	677
39	988	953	686
26	657	656	450
2	56	86	86

Tabela 9 – Resultado do 4º exemplo

Pontos p/Rota	Quantidade de Produtos	Custo Inicial	Custo Final
31	991	1912	1630
27	992	1568	1237
12	424	729	498

Tabela 10 – Resultado do 5º exemplo

Pontos p/Rota	Quantidade de Produtos	Custo Inicial	Custo Final
14	757	727	614
16	927	974	832
3	216	76	76

Com a adição deste segundo algoritmo, que não apenas contem operações custosas de comparação de índices, como também é executado logo após o primeiro, o tempo de execução no terceiro exemplo praticamente quadruplicou, alcançando a faixa dos 439.0429 mili segundos. No quarto, mais do que triplicou, alcançando os 50.40 mili segundos. Por fim, no quinto, surpreendentemente o tempo reduziu para 7.6468 mili segundos.

3.3 Discussão

Métodos sequenciais baseados em economias, como o de Clarke e Wright (1964) e suas variações, demonstram ser mais eficientes do que outros algoritmos construtivos. Porém, a distância total em suas rotas é bem maior, e podem existir rotas "excedentes"que teoricamente poderiam ser combinadas à outras, considerando apenas a capacidade dos veículos e o custo de cobrir tais rotas.

Em termos de desempenho, mesmo com a adição do algoritmo de busca local para melhoria dos resultados iniciais e com o uso de grandes exemplos, o tempo de execução do segundo programa aumentou em no máximo quatro vezes se comparado ao programa original - isso considerando o maior exemplo. Fora o curioso resultado do quinto exemplo onde ainda ocorreu uma redução no tempo de execução.

Tabela 11 – Comparando os tempos de execução

			Tempo (em mili segundos)	
Exemplo	Nº de Pontos	Nº de Rotas	CW	CW + 2-OPT
1º	10	2	0.0	1.08
2º	9	3	0.0	0.0
3º	150	5	114.312649	439.042806
4º	71	3	15.70	50.40
5º	34	3	18.665	7.6468

Observou-se que segundo algoritmo possibilita a melhoria das rotas já estabelecidas pelo primeiro ao reduzir os custos simplesmente testando a troca de posição dos pontos dentro de uma mesma rota. Curiosamente, a técnica 2-opt foi proposta anos antes do algoritmo de Clarke-Wright, mas não representa parte da composição do mesmo. Considerar que uma técnica anterior a este que é um dos algoritmos base mais eficientes na resolução do PRV até nos dias atuais foi capaz de aprimorá-lo ainda mais embasa uma observação no mínimo interessante.

4 Conclusões

O objetivo deste trabalho foi realizar uma análise comparativa de dois métodos diferentes para a resolução do Problema de Roteamento de Veículos. O primeiro é um dos algoritmos mais eficientes e conhecidos desde a época onde foi proposto, o algoritmo de Clarke e Wright (1964). Já o segundo é a busca local 2-opt (CROES, 1958) que visa melhorar o resultado gerado pelo primeiro algoritmo.

Ambos os métodos foram implementados na linguagem de programação Python com boas práticas sempre sendo consideradas. Os mesmos cinco exemplos foram testados em ambos os métodos, sendo dois exemplos pequenos com respectivamente 9 e 10 pontos a serem atendidos, um médio com 34 pontos, e dois relativamente grandes com 71 e 150 pontos, respectivamente.

A melhoria alcançada na transição de um método para o outro é notável, com eficácia nitidamente aumentada se tratando da redução de custos nas rotas definidas na execução do primeiro algoritmo e tempo de execução afetado de maneira praticamente insignificante em exemplos pequenos. Isso é surpreendente considerando a eficiência natural do primeiro algoritmo mesmo na época em que foi proposto, na década de 1960, e que ele pôde ser aprimorado ainda mais utilizando outro algoritmo que não apenas é muito simples, como foi proposto anos antes, no final da década de 1950.

Essa análise aqui realizada não apenas poderia ser aprofundada, como também poderia crescer ao considerarmos algoritmos posteriores que aprimoraram ainda mais os resultados registrados, principalmente por tratarem de problemas ou pendências fora do escopo de ambos os algoritmos vistos aqui. Não seria nenhuma novidade, mas assumindo o papel de uma revisão direta, completa e atualizada, este estudo seria muito valioso.

Referências

BENNETT, B. T.; GAZIS, D. C. School bus routing by computer. **Transportation Research**, Elsevier BV, v. 6, n. 4, p. 317–325, dez. 1972. Citado na página 23.

CHEGG. Question: Solve the following vehicle routing problem using the clarke-wright method. **Chegg**, 2018. Disponível em: https://www.chegg.com/homework-help/questions-and-answers/solve-following-vehicle-routing-problem-using-clarke-wright-method-need-determine-many-rou-q152 Citado 2 vezes nas páginas 27 e 28.

CHRISTOFIDES, N.; MINGOZZI, A.; TOTH, P. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. **Mathematical Programming**, Springer Science and Business Media LLC, v. 20, n. 1, p. 255–282, dez. 1981. Citado 2 vezes nas páginas 18 e 23.

CLARKE, G.; WRIGHT, J. W. Scheduling of vehicles from a central depot to a number of delivery points. **Operations Research**, Institute for Operations Research and the Management Sciences (INFORMS), v. 12, n. 4, p. 568–581, ago. 1964. Citado 5 vezes nas páginas 13, 18, 23, 32 e 34.

CROES, G. A. A method for solving traveling-salesman problems. **Operations Research**, Institute for Operations Research and the Management Sciences (INFORMS), v. 6, n. 6, p. 791–812, dez. 1958. Citado 5 vezes nas páginas 13, 18, 20, 23 e 34.

DANTZIG, G. B.; RAMSER, J. H. The truck dispatching problem. **Management Science**, Institute for Operations Research and the Management Sciences (INFORMS), v. 6, n. 1, p. 80–91, mar. 1959. Citado 3 vezes nas páginas 15, 16 e 23.

FISCHETTI, M.; TOTH, P.; VIGO, D. A branch-and-bound algorithm for the capacitated vehicle routing problem on directed graphs. **Operations Research**, Institute for Operations Research and the Management Sciences (INFORMS), v. 42, n. 5, p. 846–859, 10 1994. Citado na página 29.

FISHER, M. L.; JAIKUMAR, R. A generalized assignment heuristic for vehicle routing. **Networks**, Wiley, v. 11, n. 2, p. 109–124, 1981. Citado 2 vezes nas páginas 18 e 23.

GILLETT, B. E.; MILLER, L. R. A heuristic algorithm for the vehicle-dispatch problem. **Operations Research**, Institute for Operations Research and the Management Sciences (INFORMS), v. 22, n. 2, p. 340–349, abr. 1974. Citado 3 vezes nas páginas 18, 23 e 24.

GOLDEN, B. L.; MAGNANTI, T. L.; NGUYEN, H. Q. Implementing vehicle routing algorithms. **Networks**, Wiley, v. 7, n. 2, p. 113–148, 1977. Citado na página 23.

GRECO, F. (Ed.). **Traveling salesman problem**. Croatia: InTech, 2008. 202 p. ISBN 9789537619107. Citado na página 16.

KINDERVATER, G. A. P.; SAVELSBERGH, M. W. P. Vehicle routing: handling edge exchanges. In: AARTS, E. H. L.; LENSTRA, J. K. (Ed.). **Local Search in**

Combinatorial Optimization. Chichester: Princeton University Press, 2003. p. 337–360. Citado 2 vezes nas páginas 18 e 24.

KOVÁCS, L.; AGÁRDI, A.; BÁNYAI, T. Fitness landscape analysis and edge weighting-based optimization of vehicle routing problems. **Processes**, MDPI AG, v. 8, n. 11, p. 1363, out. 2020. Citado na página 16.

LARSON, R. C.; ODONI, A. R. **Urban Operations Research**. New Jersey: Prentice-Hall, 1981. 530 p. ISBN 0975914634. Citado 2 vezes nas páginas 19 e 27.

LAWLER, E. L.; LENSTRA, J. K.; KAN, A. H. G. R.; SHMOYS, D. B. **The Traveling Salesman Problem**: A guided tour of combinatorial optimization (wiley series in discrete mathematics optimization). [S.l.]: Wiley, 1991. 476 p. ISBN 9780471904137. Citado na página 15.

MOLE, R. H.; JAMESON, S. R. A sequential route-building algorithm employing a generalised savings criterion. **Journal of the Operational Research Society**, Informa UK Limited, v. 27, n. 2, p. 503–511, jun. 1976. Citado 2 vezes nas páginas 18 e 23.

MONTOYA-TORRES, J. R.; FRANCO, J. L.; ISAZA, S. N.; JIMÉNEZ, H. F.; HERAZO-PADILLA, N. A literature review on the vehicle routing problem with multiple depots. **Computers & Industrial Engineering**, Elsevier, v. 79, p. 115–129, 2015. Citado na página 24.

PICHPIBUL, T.; KAWTUMMACHAI, R. An improved clarke and wright savings algorithm for the capacitated vehicle routing problem. **ScienceAsia**, v. 38, n. 3, p. 307–318, 2012. Citado na página 24.

PIERRESELIM. **2-opt procedure**. 2009. Figura disponível sob a licença GNU Free Documentation License. Disponível em: https://commons.wikimedia.org/wiki/File:2-opt_wiki.svg. Citado na página 21.

PILLAC, V.; GENDREAU, M.; GUÉRET, C.; MEDAGLIA, A. L. A review of dynamic vehicle routing problems. **European Journal of Operational Research**, Elsevier, v. 225, n. 1, p. 1–11, 2013. Citado na página 24.

RYAN, D. M.; HJORRING, C.; GLOVER, F. Extensions of the petal method for vehicle routeing. **Journal of the Operational Research Society**, Informa UK Limited, v. 44, n. 3, p. 289–296, mar. 1993. Citado 2 vezes nas páginas 18 e 24.

THOMPSON, P. M.; PSARAFTIS, H. N. Cyclic transfer algorithm for multivehicle routing and scheduling problems. **Operations Research**, Institute for Operations Research and the Management Sciences (INFORMS), v. 41, n. 5, p. 935–946, out. 1993. Citado 2 vezes nas páginas 18 e 23.

VAN BREEDAM, A. Improvement heuristics for the vehicle routing problem based on simulated annealing. **European Journal of Operational Research**, Elsevier BV, v. 86, n. 3, p. 480–490, nov. 1995. Citado na página 24.

WILHELM, V. E. **Problema de Roteamento de Veículos**. 2015. Notas de aula. Disponível em: https://docs.ufpr.br/~volmir/PO_II_13_VRP.pdf. Citado na página 18.