UNIVERSIDADE FEDERAL DE SÃO CARLOS– UFSCar
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA– CCET
DEPARTAMENTO DE COMPUTAÇÃO– DC
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO– PPGCC

Uma Abordagem Híbrida de Metaheurística para o Problema de Roteamento de Veículos

Gabriel Ferreira Cunha

Orientador: Dr. Roberto Santos Inoue
Coorientador: Dr. Edilson Reis Rodrigues Kato

São Carlos
23 de janeiro de 2026

Gabriel Ferreira Cunha

Uma Abordagem Híbrida de Metaheurística para o Problema de Roteamento de Veículos

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Inteligência Artificial

Orientador: Dr. Roberto Santos Inoue
Coorientador: Dr. Edilson Reis Rodrigues Kato

São Carlos
23 de janeiro de 2026

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Defesa de Dissertação de Mestrado do candidato Gabriel Ferreira Cunha, realizada em 12/09/2024.

Comissão Julgadora:

Prof. Dr. Roberto Santos Inoue (UFSCar)

Prof. Dr. Samuel Lourenço Nogueira (UFSCar)

Profa. Dra. Kalinka Regina Lucas Jaquie Castelo Branco (USP)

O Relatório de Defesa assinado pelos membros da Comissão Julgadora encontra-se arquivado junto ao Programa de Pós-Graduação em Ciência da Computação.

Este trabalho é dedicado ao meu irmão Marcos Ferreira (in memoriam). Sua inspiração e grande apoio me acompanharam durante todo este processo

Agradecimentos

Agradecimentos à meu orientador, Dr. Roberto Santos Inoue [1] , por sua orientação, paciência e apoio contínuo. Suas sugestões e críticas construtivas foram fundamentais para a conclusão deste trabalho.

Expresso também minha gratidão ao meu coorientador, Dr. Edilson Reis Rodrigues Kato [2], por seu suporte técnico e científico, e por estar sempre disponível para esclarecer minhas dúvidas.

Por fim, agradeço a todos os amigos e familiares que, direta ou indiretamente, contribuíram para o meu crescimento acadêmico e pessoal durante esta jornada.

[1] <http://lattes.cnpq.br/6221209121565990>
[2] <http://lattes.cnpq.br/8517698122676145>

Resumo

O Problema de Roteamento de Veículos Capacitados (CVRP, do inglês *Capacitated Vehicle Routing Problem*) é um problema clássico em otimização combinatória, que visa otimizar as rotas de veículos com capacidade de armazenamento limitada para atender às demandas dos clientes. O objetivo principal do CVRP é encontrar uma solução viável que minimize o custo total das rotas em um grafo ponderado. Este problema é amplamente estudado devido à sua relevância em diversas aplicações práticas, como logística e distribuição.

Esta pesquisa tem como objetivo desenvolver um algoritmo utilizando Inteligência Artificial, por meio de uma solução metaheurística híbrida que integra Algoritmos Genéticos (GA, do inglês Genetic Algorithm), Otimização por Colônias de Formigas (ACO, do inglês *Ant Colony Optimization*) e Busca Tabu (TS, do inglês *Tabu Search*) para encontrar soluções ótimas ou quase ótimas para o CVRP. Os passos estabelecidos para alcançar esse objetivo incluem o desenvolvimento do algoritmo híbrido ACO+TS combinado com GA para o CVRP, a realização de experimentos computacionais utilizando instâncias de teste, a comparação dos resultados obtidos pelo algoritmo proposto com soluções ótimas conhecidas ou aquelas obtidas por outros métodos de otimização existentes, e a análise dos resultados para identificar vantagens, limitações e possíveis melhorias do algoritmo proposto.

Os resultados demonstram que o GA, ACO e TS são altamente eficazes na resolução de instâncias complexas do CVRP. Em várias instâncias testadas, as soluções encontradas foram próximas ou superaram os valores ótimos de referência, destacando a alta eficiência do método proposto.

Em conclusão, a abordagem híbrida ACO+TS combinada com AG confirmou sua eficácia na resolução de problemas complexos de roteamento de veículos, fornecendo uma base sólida para pesquisas futuras. A combinação híbrida de algoritmos contribui não apenas para a obtenção de soluções de alta qualidade, mas também para a exploração eficiente do espaço de soluções.

Palavras-chave: search problems, local search, reinforcement learning, vehicle routing, CVRP, optimization..

Abstract

The Capacitated Vehicle Routing Problem (CVRP) is a classic problem in combinatorial optimization, aimed at optimizing the routes of vehicles with limited storage capacity to meet customer demands. The primary goal of the CVRP is to find a feasible solution that minimizes the total route cost in a weighted graph. This problem is widely studied due to its relevance in various practical applications, such as logistics and distribution.

This research aims to develop a hybrid metaheuristic solution composed of Genetic Algorithms (GA), Ant Colony Optimization (ACO), and Tabu Search (TS) to find optimal or near-optimal solutions for the CVRP. The steps established to achieve this objective include the development of the hybrid algorithm ACO+TS combined with GA for the CVRP, conducting computational experiments using test instances, comparing the results obtained by the proposed algorithm with known optimal solutions or those obtained by other existing optimization methods, and analyzing the results to identify advantages, limitations, and potential improvements of the proposed algorithm.

The results demonstrate that the GA, ACO, and TS are highly effective in solving complex instances of the CVRP. In several tested instances, the solutions found were close to or exceeded the reference optimal values, highlighting the high efficiency of the proposed method. In conclusion, the hybrid ACO+TS approach combined with GA has confirmed its effectiveness in solving complex vehicle routing problems, providing a solid foundation for future research. The hybrid combination of algorithms contributes to obtaining high-quality solutions and efficiently exploring the solution space.

Keywords: search problems, local search, reinforcement learning, vehicle routing, CVRP, optimization..

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ACO	Ant Colony Optmization
ACS-BSO	Ant Colony System - Bee Swarm Optimization
GA	Genetic Algorithm
HGA	Hybrid Genetic Algorithm
BACO	Bilevel Ant Colony Optmization
BC	Branch and Cut
CC	Capacity Constraints
CEVRP	Capacitated Electric Vehicle Routing Problem
CD	Centro de Distribuição
CE	Centro de Estocagem
CRVPLIB	Capacitated Vehicle Routing Problem Library
CVRP	Capacitated Vehicle Routing Problem
DRL	Deep Reiforcement Learningh
EVRP	Electric Capacitated Vehicle Routing Problem
FRVCP	Fixed Route Vehicle Charging Problem
GA	Genetic Algorithm
GRASP	Greedy Randomized Adaptive Search Procedure
GTS	Granular tabu search solution
ILC	Iterated LocalSearch

IPC	Infeasible Path Constraints
LSVRP	large-scale vehicle routing problems
MMAS	Max-Min Ant System
PDP	Pickup and Delivery Problem
PDPSLT	Pickup and Delivery Problem with Split Loads and Transshipments
PRC	Predecessor or Successor Constraints
PSTTRPTW	Profitable Single Truck and Trailer Routing Problem with Time Windows
SA	Simulated Annealing
SCC	Strengthened Capacity Constraints
SEC	Subtour Elimination Constraints
TS	Tabu Search
TSP	Travelling Salesman Problem
RTR	Record-to-record travel
VNS	Variable Neighborhood Search
VRP	Vehicle Routing Problem
VRTR	Variable-length neighbor list record-to-record travel solution

Sumário

Capítulo 1

INTRODUÇÃO

O Problema de Roteamento de Veículos com Capacidade (CVRP, do inglês *Capacitated Vehicle Routing Problem*) é um problema amplamente conhecido em otimização combinatória, que visa otimizar rotas de veículos com capacidade de armazenamento limitada para atender às demandas específicas dos clientes (GUEDES; LEITE; ALOISE, 2009). O principal objetivo do CVRP é encontrar rotas viáveis que minimizem os custos, garantindo que os veículos entreguem as mercadorias de acordo com os requisitos de carga dos clientes (MINGPRASERT; MASUCHUN, 2017). Isso envolve, muitas vezes, a busca pela rota de menor custo em um grafo ponderado dentro de um ciclo Hamiltoniano, segundo Laporte (1992).

O CVRP é especialmente relevante na logística de comércio eletrônico, onde o processo de entrega é dividido em três fases: a primeira milha, a milha intermediária e a última milha (GALINDO et al., 2020). Na primeira milha, os pacotes são armazenados em centros de distribuição (CD) e, em seguida, agrupados e enviados para centros de expedição (CE). A milha intermediária consiste na distribuição dos pacotes dos CDs para os CEs, utilizando caminhões ou aviões, onde os pacotes são agrupados para regiões específicas. Finalmente, a última milha envolve a entrega ao consumidor final, utilizando veículos diversos, como carros, motocicletas ou vans, de acordo com o volume e as especificidades dos pacotes.

A última milha é reconhecida como a fase mais complexa e de custo mais elevado, representando cerca de 50% do custo total de entrega. Essa complexidade é devido à necessidade de otimização das rotas e agrupamento de pacotes para minimizar custos (SOUZA et al., 2020). Em um cenário urbano, o custo anual da logística na última milha é estimado em aproximadamente 70 bilhões de euros (LINDHOLM; WIEDING, 2012), o que leva as empresas de logística a buscarem soluções inovadoras para reduzir esses custos, sem comprometer a qualidade do serviço, e ao mesmo tempo minimizar os

impactos ambientais e econômicos de suas operações (SOUZA et al., 2020).

Para enfrentar esses desafios, o CVRP visa otimizar a distribuição ao longo de uma rota, respeitando as restrições de capacidade de cada veículo com base no volume e peso dos pacotes (GALINDO et al., 2020). No entanto, a complexidade computacional do CVRP torna-se evidente em problemas de grande escala, onde soluções exatas são computacionalmente viáveis apenas para pequenas instâncias (GUEDES; LEITE; ALOISE, 2009). Assim, para problemas maiores, é essencial utilizar abordagens eficientes que permitam encontrar soluções próximas ao ótimo, reduzindo a distância percorrida e o tempo de rota, especialmente na última milha.

A complexidade NP-difícil do CVRP para grandes instâncias (GALINDO et al., 2020; LI; GOLDEN; WASIL, 2005) motivou o desenvolvimento de uma variedade de algoritmos de otimização, incluindo técnicas de inteligência artificial e metaheurísticas, como Algoritmos Genéticos (GA) (DENG et al., 2012), Agrupamento Fuzzy de C-means (ELNEIMA; SALIH, 2021), Otimização por Colônia de Formigas (ACO) (JIA; MEI; ZHANG, 2021; HENDRAWAN; BAIZAL; WULANDARI, 2024), Busca Tabu (TS) (LI et al., 2021; OBAID, 2018; MACHADO et al., 2021), Aprendizado por Reforço Profundo (LI et al., 2021), Google OR-Tools e Lin-Kernighan-Helsgaun (LKH) (TAILLARD; HELSGAUN, 2019). Esses algoritmos empregam técnicas heurísticas e metaheurísticas que permitem encontrar soluções eficazes em um tempo de processamento aceitável, especialmente para atender às demandas de entrega rápida.

Para validar esses algoritmos, benchmarks de CVRP são amplamente utilizados. Machado et al. (2021) forneceram soluções para problemas de pequeno e médio porte, enquanto Li, Golden e Wasil (2005) desenvolveram instâncias com até 1.200 clientes, usadas frequentemente para testar e comparar diferentes abordagens. Dentre essas instâncias, destacam-se Li_22, Li_25 e Li_26, que são amplamente utilizadas como referência para verificar a eficácia de novas metodologias em situações com alta complexidade.

As abordagens híbridas, que combinam diferentes algoritmos, têm se mostrado vantajosas na obtenção de soluções de alta qualidade e eficiência computacional (TOTH et al., 2015). Por exemplo, Machado et al. (2021) propuseram uma matheurística híbrida, utilizando uma abordagem hierárquica de dois estágios para resolver o CVRP, que integra o Procedimento de Busca Adaptativa Gulosa e Aleatória (GRASP, do inglês *Greedy Randomized Adaptive Search Procedure*) com métodos exatos e Busca em Vizinhança Variável (VNS, do inglês *Variable Neighborhood Search*). Essa abordagem demonstrou desempenho competitivo em problemas de pequeno e médio porte, otimizando as rotas para reduzir o custo de entrega e o tempo de processamento.

Diante desse contexto, o presente trabalho visa desenvolver uma solução híbrida para o CVRP, combinando Otimização por Colônia de Formigas (ACO) com Busca Tabu (TS) para melhorar a eficiência na busca de soluções. Essa combinação permite refinar a probabilidade de construção de soluções e realizar atualizações iterativas de

feromônios, enquanto a TS amplia o espaço de busca para evitar mínimos locais. Além disso, o uso de Algoritmos Genéticos (GA) para gerar a população inicial para o ACO visa otimizar ainda mais a eficiência e a qualidade das soluções. Com essa abordagem, o trabalho conseguiu melhorar os resultados em algumas instâncias de benchmark propostas por Li, Golden e Wasil (2005), especialmente as instâncias Li_22, Li_25 e Li_26, que apresentaram desempenho superior em relação aos resultados reportados anteriormente.

1.1 Organização do Trabalho

Este trabalho está organizado da seguinte forma: No Capítulo 2 serão apresentados os conceitos sobre modelo de entrega, problema de roteamento de veículos e suas restrições e, ainda, demostrará os principais métodos de solução para o CVRP, apresentando os conceitos de heurísticas, meta-heurística e o funcionamento do algoritmo colonia de formigas. No Capítulo 3 fornece uma revisão das pesquisas relevantes, destacando contribuições e metodologias principais que justificam a relevância e a necessidade do estudo atual. O Capítulo 4 introduz a proposta de pesquisa, detalhando a abordagem híbrida que combina os algoritmos ACO e TS com o GA, incluindo sua integração e implementação. O Capítulo 4.2 apresenta e discute os resultados experimentais, avaliando o desempenho do método híbrido proposto em comparação com soluções existentes. Finalmente, no Capítulo 5 resume as conclusões obtidas a partir desta pesquisa, discutindo as implicações dos resultados e possíveis áreas para trabalhos futuros.

Capítulo 2

CONCEITOS

2.1 Modelo de Entrega

Dentre os diversos modelos existentes, o uso de motoristas independentes é a forma mais usada para o procedimento de entregas em última etapa. Esse modelo, utilizado regularmente por grandes empresas de entregas, como Loggi, Uber e Rappi, é conhecido como economia compartilhada, no qual os motoristas se cadastram em uma plataforma online e recebem uma lista de pacotes a serem entregues, sem necessidade de retorno ao depósito. Em contrapartida, nos modelos tradicionais, as empresas são responsáveis por todo o processo de entrega até a rota final, a exemplo dos Correios. Para ambos os modelos, os veículos devem buscar os pacotes no centro de estocagem (CE) ou no centro distribuição (CD) e devem seguir a rota da entrega final, a qual é definida a partir da lista de pacotes a serem distribuídos (GALINDO et al., 2020).

A Figura 1 mostra o funcionamento do modelo de distribuição no formato de economia compartilhada utilizado pela empresa de logística Loggi, o qual indica a alocação de cada veículo e define a menor rota de entrega a ser executada com base na ordem dos pacotes.

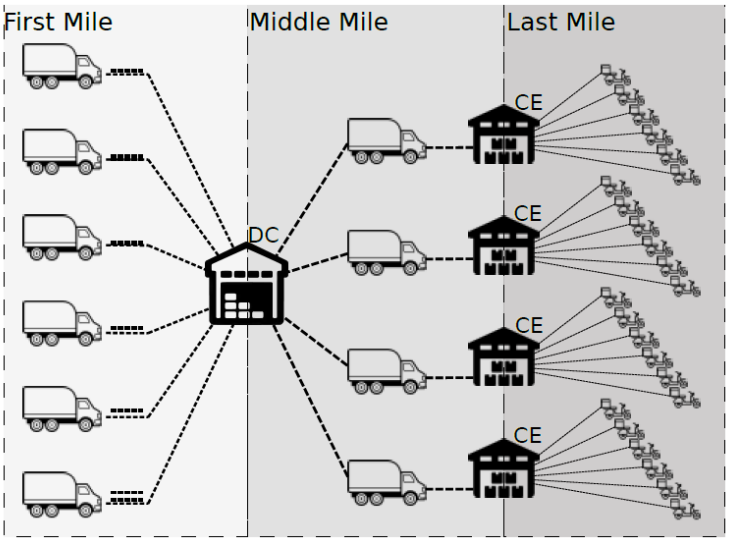

Figura 1 – Modelo de entregas da Loggi
(GALINDO et al., 2020)

Para tornar o modelo sustentável, as empresas de logística têm enfrentado inúmeros desafios. Dentre eles estão a redução do tempo de processamento e a complexidade para realizar o roteamento de pacotes, agrupando-os em cargas e gerando rotas para serem entregues aos consumidores finais. Nessa última, os motoristas independentes buscam os pacotes nos CEs para realizar as entregas aos consumidores finais (GALINDO et al., 2020).

2.2 Problema de Roteamento de Veículos

O Problema de Roteamento de Veículos Capacitados (CVRP) é uma variante do Problema de Roteamento de Veículos (VRP) que incorpora restrições de armazenamento. Cada veículo tem um limite na quantidade de pacotes que pode transportar em uma rota, o qual depende diretamente de sua capacidade de armazenamento. Essa restrição influencia diretamente o principal objetivo do CVRP: encontrar a rota mais curta entre o depósito e vários destinos. As rotas são calculadas com base na quantidade e capacidade dos veículos disponíveis, de modo que a distância percorrida por cada veículo e, consequentemente, a eficiência operacional geral, como o consumo de combustível e as despesas com frete, sejam otimizadas, o que é necessário para operações logísticas eficientes. No CVRP, várias restrições são definidas: cada veículo tem um limite na quantidade de pacotes que pode transportar para uma rota dada. A capacidade é determinada pelo tipo de veículo utilizado. Consequentemente, o CVRP visa encontrar a rota mais curta entre o depósito e vários destinos de entrega e deve retornar ao depósito (JIA; MEI; ZHANG, 2021), (LAPORTE, 1992).

Para calcular as rotas usando os veículos disponíveis de modo que a distância percorrida por cada um seja minimizada, os veículos utilizados para entregas na última milha devem ter uma capacidade máxima de carga. Restrições adicionais, como peso,

volume, número de pacotes e janelas de tempo, podem ser adicionadas como variações do Problema de Roteamento de Veículos (VRP).

As restrições para o CVRP são definidas da seguinte forma. Para esta pesquisa, focamos no modelo de minimização de custo, no qual as restrições são definidas para minimizar a rota enquanto se adere às seguintes regras: garantir que cada rota seja atendida por apenas um veículo, que nenhum veículo exceda sua capacidade e que a rota de cada veículo comece e termine no depósito.

De acordo com (ZULFIQAR; ISNANTO; NURHAYATI, 2018), o CVRP pode ser representado por um modelo matemático que define suas restrições e parâmetros (FAIZ; KRICHEN; INOUBLI, 2014).

Os parâmetros que definem o problema são:

- n Conjunto de veículos e depósitos,
- m Número de veículos,
- C Capacidade de cada veículo,
- D_i Demanda do cliente i,
- d_{ij}^k Distância entre o cliente i e o cliente j para o veículo k, e
- $x_{ij}^k = \begin{cases} 1 & \text{se o veículo } k \text{ viaja de } i \text{ para } j \\ 0 & \text{caso contrário} \end{cases}$
- K Número total de veículos utilizados.

As restrições são definidas como:

1. Minimizar a distância total percorrida enquanto se adere às restrições do sistema

$$\text{Minimizar } (f(x)) = \sum_{i=1}^{n} \sum_{j=i}^{n} \sum_{k=1}^{m} d_{ij} x_{ij}^k. \tag{1}$$

2. Garantir que cada ponto na rota seja visitado por exatamente um veículo.

$$\sum_{k=1}^{m} \sum_{j=0}^{n} x_{ij}^k = 1 \qquad \text{para } j = 0, \ldots, i-1, i+1, \ldots, n. \tag{2}$$

3. Definir as restrições de capacidade dos veículos.

$$\sum_{j=0}^{n} D_j \left(\sum_{i=0}^{n} x_{ij}^k \right) \leq C \qquad \text{para } k = 1, \ldots, m,\ i \neq j. \tag{3}$$

4. Garantir que os veículos comecem e terminem no depósito.

$$\sum_{j=0}^{n} x_{j0}^{k} \leq 1 \qquad \text{para } k = 1, \ldots, m, \tag{4}$$

$$\sum_{i=0}^{n} x_{0i}^{k} \leq 1 \qquad \text{para } k = 1, \ldots, m. \tag{5}$$

Para abordar o Problema de Roteamento de Veículos Capacitados (CVRP), várias equações chave são utilizadas para modelar as restrições e objetivos. A Equação (1) visa minimizar a distância total percorrida por todos os veículos, o que é essencial para otimizar as rotas. A Equação (2) garante que cada cliente seja visitado exatamente uma vez, somando as variáveis de decisão para todos os veículos e destinos. Esta restrição garante que nenhum cliente seja omitido ou visitado mais de uma vez. A Equação (3) impõe restrições de capacidade aos veículos, garantindo que a demanda total atendida por cada veículo não exceda sua capacidade. Isso é crucial para manter a viabilidade operacional e assegurar as restrições do problema. Finalmente, as Equações (4) e (5) garantem que cada veículo comece e termine sua rota no depósito. Essas equações, em conjunto, asseguram que os veículos operem dentro das restrições definidas e otimizem efetivamente o processo de roteamento (GALINDO et al., 2020).

Dadas essas restrições, existem várias abordagens para resolver o problema. Vários autores utilizam técnicas metaheurísticas para explorar o espaço de busca e encontrar soluções eficientes, considerando fatores como as restrições definidas. Essas abordagens visam melhorar a eficiência e a otimização do CVRP.

2.3 Heurística

Uma das abordagens utilizadas para encontrar soluções aproximadas ou satisfatórias para problemas complexos quando os métodos exatos são inviáveis ou demorados é a heurística. Baseada em estratégias práticas, intuições, regras empíricas ou experiências anteriores, a heurística busca, ao invés de garantir uma solução ótima, encontrar soluções que se aproximem o suficiente do resultado desejado, em tempo e recurso aceitáveis Papadimitriou e Steiglitz (1998).

A palavra "heurística" tem sua origem no grego, mais precisamente no termo "heuriskein", que significa "descobrir" ou "encontrar". Essa origem etimológica reflete diretamente o propósito e a natureza das heurísticas como métodos de solução aproximada para problemas complexos. Ao utilizar estratégias práticas, intuições e regras empíricas, as heurísticas buscam descobrir soluções eficientes que podem ser encontradas de maneira mais rápida e efetiva do que os métodos exatos. A abordagem tem sido amplamente aplicada em diversas áreas do conhecimento, abrangendo a otimização, a inteligência artificial e a resolução de problemas do mundo real. A palavra "heurística"

captura, portanto, a essência de um processo dinâmico e exploratório. Martí e Reinelt (2022).

Tendo em vista as frequentes restrições dos métodos exatos no oferecimento de soluções satisfatórias a problemas complexos, cresceu a necessidade de criar e aperfeiçoar técnicas heurísticas para lidar com problemas considerados NP-Difícil. Essas abordagens visam encontrar soluções aproximadas e eficientes, mesmo que não garantam a solução ótima. O foco desses métodos é utilizar estratégias práticas, intuição e experiência para explorar o espaço de busca do problema, alcançando resultados satisfatórios dentro de limitações de tempo e recursos computacionais Toth et al. (2015).

2.4 Meta-heurística

Se o objetivo de uma heurística, enquanto método computacional, é encontrar uma solução considerada aceitável, viável e ótima para um determinado problema, uma meta-heurística consiste em uma abordagem geral composta por um conjunto de regras que combinam métodos heurísticos, visando aumentar a eficiência e a eficácia na exploração do espaço de busca(GOLDBARG; LUNA, 2016).

Tendo em vista que o prefixo "meta" se refere a uma posição posterior, pode-se compreender que a meta-heurística é a combinação de métodos heurísticos. Pode-se entendê-la, portanto, como um conjunto de métodos heurísticos gerais que orientam o desenvolvimento de heurísticas específicas, ou seja, estratégias mais abrangentes para o desenvolvimento de métodos menores.

As meta-heurísticas têm sido amplamente aplicadas em diversos domínios, abrangendo problemas de otimização. Sua eficácia está relacionada à capacidade de superar a busca local, explorando, de forma inteligente, o espaço de soluções em busca de resultados melhores. Essas técnicas têm contribuído significativamente para a resolução de problemas complexos, nos quais os métodos exatos podem ser inviáveis ou demandar um tempo excessivo de processamento. Além disso, as meta-heurísticas continuam sendo objeto de pesquisa e desenvolvimento, com esforços constantes para melhorar seu desempenho e adaptabilidade a diferentes contextos de otimização (BLUM; ROLI, 2003).

2.4.1 Colônia de Formigas "*Ant Colony*"

Inspirada no comportamento de forrageamento de formigas, a Colônia de Formigas é um algoritmo meta-heurístico proposto por Dorigo e L.M. (1997), o qual tem sido amplamente utilizado para resolver problemas de otimização, como o VRP.

Nas colônias de formigas reais, as formigas individuais deixam rastros de feromônios enquanto procuram por alimentos. Esses feromônios funcionam como sinais químicos que indicam o caminho percorrido pela formiga. Quando outras formigas encontram esses rastros de feromônio, há uma tendência de seguirem essas trilhas, aumentando

assim a probabilidade de encontrar alimentos de maneira eficiente. No algoritmo Colônia de Formigas, cada formiga virtual representa uma solução em potencial para o problema em questão. As formigas constroem soluções de forma iterativa, fazendo escolhas probabilísticas baseadas em critérios como a quantidade de feromônio presente nas arestas do grafo que representa o problema (DORIGO; L.M., 1997).

À medida que as formigas constroem suas soluções, elas depositam feromônio nas arestas que percorreram. O feromônio evaporará gradualmente com o tempo, simulando a redução da intensidade do rastro de feromônio na natureza. Dessa forma, soluções melhores receberão mais feromônio, aumentando a probabilidade de serem selecionadas por outras formigas. Ao longo das iterações do algoritmo, a concentração de feromônio nas arestas é atualizada com base na qualidade das soluções encontradas. Com o tempo, o algoritmo converge para uma solução ótima ou próxima do ótimo (DORIGO; L.M., 1997).

2.4.2 Algoritmo Genético

A exemplo da Colônia de Formigas, o Algoritmo Genético também busca inspiração na biologia. Proposta por Holland (1975), essa técnica tem sido amplamente utilizada para resolver problemas de otimização. O algoritmo genético opera com uma população de soluções candidatas, representadas geralmente por cromossomos ou indivíduos. Cada cromossomo contém um conjunto de genes que codificam uma solução potencial. Esses genes podem representar características específicas ou parâmetros do problema em questão.

O algoritmo genético segue um processo iterativo que simula a evolução natural, a partir de uma população inicial de soluções aleatórias. Em cada iteração, chamada de geração, a população passa por uma série de operações: seleção, cruzamento (crossover) e mutação.

Na seleção, indivíduos mais aptos têm maior probabilidade de serem selecionados e se reproduzem com base em um método de aptidão que avalia a qualidade de cada solução. Essa seleção favorece soluções melhores, assim como na seleção natural.(HOLLAND, 1975)

O cruzamento é uma operação que combina informações de dois ou mais cromossomos para gerar novos descendentes. Esse processo simula o cruzamento genético na reprodução biológica. Pode ser realizado de várias maneiras; no ponto de corte, por exemplo, os genes dos pais são trocados em uma posição específica. (HOLLAND, 1975)

A mutação é uma operação que introduz pequenas alterações aleatórias em um cromossomo, simulando a ocorrência de mutações genéticas. Essas mutações podem ajudar a explorar o espaço de busca de soluções e evitar a convergência prematura para uma solução sub ótima. Após a aplicação dessas operações, uma nova geração é formada e o processo se repete até que um critério de parada seja atingido, como um número

máximo de gerações ou a convergência para uma solução satisfatória (HOLLAND, 1975).

Capítulo 3

REVISÃO BIBLIOGRÁFICA

Os algoritmos exatos clássicos para o problema de roteamento de veículos com limitação de Carga Toth et al. (2015) define o Problema de Roteamento de Veículos com limitação de Carga (CVRP) como sendo uma extensão do conhecido Problema do Caixeiro Viajante (TSP, do inglês *Travelling salesman problem*), que exige a determinação de um percurso fechado em um grafo, no qual cada vértice é visitado exatamente uma vez, ou seja, um circuito hamiltoniano com custo mínimo visitando somente uma vez um determinado conjunto de pontos. Entretanto, o CVRP está longe de ser resolvido satisfatoriamente conforme a análise dos autores que abrange mais de três décadas de pesquisa e examina as principais abordagens de métodos diretos à busca e buscas lineares exatas. Os autores apresentam várias técnicas utilizadas para encontrar soluções ótimas, algumas delas utilizando algoritmos exatos como os algoritmos de Ramificação e Corte e outros utilizam algoritmos baseados em heurística e meta-heurística para encontrar uma solução ótima. Desta feita, nos tópicos a seguir serão apresentadas algumas aplicações dos métodos e alguns dos algorítimos citados.

Algoritmos de Ramificação e Corte visam introduzir novas desigualdades válidas ausentes no modelo original. O autor (TOTH et al., 2015) apresenta quatro categorias de desigualdades: desigualdades de capacidade arredondadas, restrições generalizadas de capacidade, desigualdades combinadas e desigualdades hipotéticas. Além disso, foi utilizada uma heurística baseada em busca tabu para estabelecer um limite superior inicial e atualizá-lo com base nas soluções fracionárias encontradas durante a execução do algoritmo. Por fim, o autor investigou diferentes estratégias de ramificação, considerando as restrições do problema.

O trabalho de Cruz e Cunha (2023) trás os resultados para resolução do problema de Roteamento de Caminhão e Reboque com Janelas de Tempo (PSTTRPTW, do inglês *Profitable Single Truck and Trailer Routing Problem with Time Windows*). Este problema

é definido como um conjunto de requisitos que, em comparação com o TSP e o VRP, e ampliação do conjunto de restrições operacionais que precisam ser aplicadas para projetar rotas únicas ou múltiplas viáveis. Portanto, dependendo de qual característica do problema é considerada, diferentes famílias ou classes de problemas de roteamento podem ser diretamente relacionadas ao PSTTRPTW. No que se refere ao tema, Cruz e Cunha (2023) mostram o estudo sobre os benefícios na resolução de problemas complexos PSTTRPTW, por meio da realização de testes de pré-processamento e desigualdades válidas, e com intuito de melhorar a eficiência dos algoritmos utilizados, o teste foi aplicado ao algorítimo (BC, do inglês *Branch cut*) baseado na formulação básica do problema e outro no modelo reforçado, baseado em modelagem matemática. Nesse mesmo sentido, Cruz e Cunha (2023) identificou que reduziram consideravelmente o número de variáveis necessárias para representar o problema, alcançando uma redução média de cerca de 30%.

Uma rota viável para o PSTTRPTW é definida como uma rota, começando e terminando no depósito principal, visitando o cliente no máximo uma vez. Os vértices principais, por outro lado, podem ser visitados mais de uma vez. Dada uma rota atribuída a um veículo que visita um cliente, o tempo em que o veículo chega em nesta rota é dado pela soma dos tempos de viagem de todos os pontos que aparecem antes, os tempos de serviço e calculado com base nos tempos de espera de todos os clientes, e os tempos de acoplamento, transferência de carga e desacoplamento que ocorrem antes na rota. Além disso, observe que o tempo de viagem para pontos em sub-rotas é calculado usando a velocidade do caminhão, em vez da velocidade do veículo completo, (CRUZ; CUNHA, 2023).

Segundo Cruz e Cunha (2023) o modelo com desigualdades válidas e possível fortalecer e melhorar as relaxações de Programação Linear. A otimização permitiu que ele comparasse as relaxações fornecidas por ambos os modelos, concluindo que o algoritmo BC reforçado apresentava relaxação, em média, 25% mais nítidas do que o modelo básico com instâncias com até 40 pontos o algoritmo foi capaz de encontrar soluções viáveis rapidamente. No entanto, para instâncias maiores, teve dificuldade em encontrar números inteiros viáveis.

Wolfinger e Salazar-González (2021) apresenta um problema de coleta e entrega *Pickup and Delivery Problem* (PDP), sendo necessário, para esse tipo de problema, o roteamento de uma frota de veículos para atender a um conjunto de solicitações dos clientes. Cada uma das solicitações envolve a coleta de uma carga em um local de origem e sua entrega em um local de destino. É permitido transportar simultaneamente várias solicitações no mesmo veículo, desde que a carga combinada não exceda sua capacidade. Cada origem e destino devem ser visitados exatamente uma vez. As rotas dos veículos devem começar e terminar nos depósitos e devem obedecer às restrições, apresentando o problema base. O problema mencionado é uma versão generalizada do PDP, chamado Problema de Coleta e Entrega com Cargas Divididas e Transbordos *Pickup and Delivery*

Problem with Split Loads and Transshipments (PDPSLT), que descreve como foi realizado a modelagem do problema, e o algoritmo *branch-and-cut* para resolver o problema PDPSLT.O foco principal está nos procedimentos heurísticos de separação usados para identificar desigualdades válidas violadas, que inicia resolvendo a relaxação linear do modelo. Se a solução encontrada for inteira para as variáveis x, y e z, então será encontrada uma solução ótima para o PDPSLT e o algoritmo termina. Caso contrário, inicia-se uma árvore de *branch-and-bound* para ser explorada. Em cada nó dessa árvore, será gerado uma desigualdade violada utilizando a heurística de separação descrita a seguir. Para melhorar o entendimento e facilitar a descrição dos procedimentos, o autor apresenta algumas notações adicionais. Dada uma solução de uma relaxação linear da formulação matemática definido como $[e*, f*, s*, x*, y*]$ deixar $Gk* = (Vk*, Ak*)$ seja um grafo auxiliar para o veículo $k \in K$ no qual $Vk* = v \in V|yk*i > 0$ é o conjunto de vértices, $Ak* = a \in Ak|xk*a > 0$ é o arco definido, e $xk*$ a é a capacidade de cada arco $a \in Ak*$. como vértice inicial a partir do ponto no qual se construi uma árvore de caminhos que consiste apenas em arcos do conjunto $Ak*$. Cada caminho é estendido enquanto o fluxo total nos arcos no caminho P é estritamente maior que $|P| - 2$ é o caminho não alcançou o depósito final do veículo $k-$. Sempre que um caminho inviável é detectado, a restrição de caminho inviável correspondente é gerada, como o artigo mostra como solução com bons resultados para o problema de instâncias com até 8 pontos(WOLFINGER; SALAZAR-GONZáLEZ, 2021).

Os resultados demonstram as desigualdades válidas utilizadas, as quais são: restrições de eliminação de sub rota (SEC, do inglês *Subtour Elimination Constraints*); Restrições de caminho inviáveis (IPC, do inglês *Infeasible Path Constraints*); restrições de capacidade (CC, do inglês *Capacity Constraints*); restrições de capacidade reforçadas (SCC, do inglês *Strengthened Capacity Constraints*); restrições predecessoras ou sucessoras (PRC, do inglês*Predecessor or Successor Constraints*). As tabelas 1 e 2 demonstram os resultados obtidos em (WOLFINGER; SALAZAR-GONZáLEZ, 2021), baseadas em 36 instâncias que o autor realizou da análise das combinações de desigualdades válidas a representado pela coluna(B‘&’C), a coluna (*Compact*) representa a execução sem o uso de desigualdades, os limites inferiores nos nós raízes, expresso como uma porcentagem do nó superior mais conhecido vinculado, mostrado na coluna (LB), a coluna (time) apresenta tempo médio necessário para resolver a instância de otimização em minutos, a coluna (#) são os cortes gerados. No entanto, algumas instâncias não puderam ser resolvidas até a otimalidade, resultando em uma lacuna percentual que representa na coluna (GAP). Esses dados fornecem uma visão abrangente do desempenho das diferentes abordagens utilizadas, permitindo uma análise comparativa para identificar a eficácia das desigualdades válidas em relação à abordagem *Compact*.

Name	Compact				SEC				IPC				CC			
	LB	Time	Gap	#	LB	Time	Gap	#	LB	Time	Gap	#	LB	Time	Gap	#
5-2-1-3-0	76.60	62.38	0.00	0	76.60	1440.00	13.15	50033	76.60	818.86	0.00	351	82.21	472.50	0.00	35
5-2-2-3-0	76.60	248.74	0.00	0	76.60	1440.00	14.65	55953	76.60	1440.00	5.16	255	82.21	815.24	0.00	29
5-2-3-3-0	76.93	495.67	0.00	0	76.60	1440.00	16.23	43317	76.60	1440.00	6.91	155	82.23	1440.00	3.12	32
5-3-1-3-0	77.22	522.90	0.00	0	76.84	1440.00	11.23	63500	77.24	1440.00	6.96	103	78.52	1440.00	6.49	24
5-3-2-3-0	76.92	1440.00	4.64	0	77.09	1440.00	15.02	62473	76.84	1440.00	9.24	42	78.64	1440.00	7.19	18
5-3-3-3-0	76.84	1440.00	5.43	0	77.79	1440.00	10.88	64560	77.84	1440.00	12.22	100	78.52	1440.00	9.95	20
6-2-1-3-0	79.80	1440.00	1.82	0	79.84	1440.00	10.89	54617	79.65	1440.00	9.55	147	79.81	1440.00	7.37	38
6-2-2-3-0	80.58	1440.00	3.41	0	79.58	1440.00	12.32	63204	80.03	1440.00	9.46	165	79.54	1440.00	9.13	38
6-2-3-3-0	79.67	1440.00	5.00	0	79.67	1440.00	15.17	64042	79.54	1440.00	10.76	70	79.67	1440.00	10.40	41
6-3-1-3-0	81.61	216.95	0.00	0	80.75	1440.00	11.18	53824	80.75	1440.00	6.51	153	82.74	1440.00	3.18	31
6-3-2-3-0	80.75	1368.27	0.00	0	80.75	1440.00	13.02	49075	83.29	1440.00	8.30	52	82.74	1440.00	7.25	32
6-3-3-3-0	80.78	1440.00	2.65	0	80.78	1440.00	12.61	47518	82.62	1440.00	8.33	102	82.77	1440.00	7.98	27
7-2-1-3-0	82.43	1440.00	7.76	0	82.43	1440.00	14.24	51357	80.05	1440.00	11.59	1140	87.20	1440.00	7.08	67
7-2-2-3-0	79.91	1440.00	8.79	0	79.93	1440.00	16.79	66119	79.91	1440.00	12.19	160	86.09	1440.00	8.58	71
7-2-3-3-0	79.91	1440.00	9.95	0	79.92	1440.00	17.71	71835	79.91	1440.00	13.44	424	85.83	1440.00	9.04	71
7-3-1-3-0	68.38	1440.00	15.28	0	68.38	1440.00	25.52	71089	68.37	1440.00	20.68	591	68.71	1440.00	22.06	44
7-3-2-3-0	68.38	1440.00	15.68	0	68.38	1440.00	26.38	74019	68.38	1440.00	23.80	330	68.71	1440.00	21.27	40
7-3-3-3-0	68.38	1440.00	17.43	0	68.38	1440.00	28.50	72136	68.38	1440.00	23.20	287	68.71	1440.00	23.82	32
8-2-1-3-0	77.47	1440.00	15.68	0	77.66	1440.00	20.20	55698	76.15	1440.00	18.56	334	85.24	1440.00	11.28	117
8-2-2-3-0	75.09	1440.00	18.17	0	75.09	1440.00	21.91	62973	75.09	1440.00	20.68	186	80.03	1440.00	14.30	109
8-2-3-3-0	77.64	1440.00	15.73	0	77.64	1440.00	19.39	61056	77.64	1440.00	18.52	298	84.80	1440.00	11.55	105
8-3-1-3-0	76.23	1440.00	13.63	0	76.23	1440.00	22.23	56362	79.38	1440.00	16.89	139	78.39	1440.00	17.05	92
8-3-2-3-0	76.23	1440.00	14.29	0	76.23	1440.00	21.78	55084	76.23	1440.00	19.15	67	78.36	1440.00	16.86	94
8-3-3-3-0	79.40	1440.00	14.36	0	76.23	1440.00	21.43	75780	76.23	1440.00	20.47	52	78.37	1440.00	17.90	87
9-2-1-3-0	73.77	1440.00	16.22	0	73.77	1440.00	25.21	83106	73.77	1440.00	20.73	61	78.46	1440.00	17.37	84
9-2-2-3-0	73.77	1440.00	16.94	0	73.78	1440.00	24.72	119287	73.77	1440.00	21.51	79	78.50	1440.00	17.76	81
9-2-3-3-0	73.43	1440.00	17.87	0	73.43	1440.00	25.44	49824	73.25	1440.00	22.00	107	78.10	1440.00	18.47	78
9-3-1-3-0	77.67	1440.00	17.20	0	76.32	1440.00	22.39	41893	77.78	1440.00	20.29	67	79.16	1440.00	17.82	92
9-3-2-3-0	80.86	1440.00	14.00	0	80.75	1440.00	18.34	57218	79.41	1440.00	17.65	53	82.52	1440.00	13.57	83
9-3-3-3-0	78.33	1440.00	15.74	0	78.33	1440.00	20.27	39148	78.33	1440.00	19.19	21	81.33	1440.00	16.05	69
10-2-1-3-0	79.87	1440.00	16.74	0	79.78	1440.00	20.06	48800	74.83	1440.00	21.13	56	82.41	1440.00	15.33	96
10-2-2-3-0	74.84	1440.00	18.12	0	76.49	1440.00	22.61	42837	78.63	1440.00	19.17	124	80.81	1440.00	16.93	95
10-2-3-3-0	76.32	1440.00	17.52	0	79.42	1440.00	20.27	36396	74.83	1440.00	22.03	186	82.34	1440.00	15.51	85
10-3-1-3-0	81.65	1440.00	13.12	0	81.83	1440.00	16.67	21393	75.30	1440.00	19.50	74	81.90	1440.00	14.33	77
10-3-2-3-0	75.36	1440.00	16.11	0	75.37	1440.00	20.84	16679	79.57	1440.00	17.16	135	78.85	1440.00	16.82	89
10-3-3-3-0	57.04	1440.00	37.86	0	57.04	1440.00	42.78	20481	61.26	1440.00	36.81	62	59.60	1440.00	37.29	88
Average	76.57	1280.97	13.57	0	76.55	1440.00	19.22	56186	76.50	1422.75	16.28	187	79.56	1395.77	13.83	64

Tabela 1 – Contribuição de desigualdades válidas (resultados individuais)(WOLFINGER; SALAZAR-GONZáLEZ, 2021)

Name	SCC				PRC				B&C			
	LB	Time	Gap	#	LB	Time	Gap	#	LB	Time	Gap	#
5-2-1-3-0	84.26	258.85	0.00	225	83.04	1440.00	6.93	24276	90.38	1440.00	5.54	9836
5-2-2-3-0	83.92	911.96	0.00	247	83.04	1440.00	8.85	23298	90.12	1440.00	7.49	10639
5-2-3-3-0	84.27	1440.00	1.64	233	83.06	1440.00	12.33	25785	90.40	1440.00	7.16	19230
5-3-1-3-0	80.91	1440.00	6.51	200	90.97	1440.00	5.27	12559	93.71	1440.00	3.63	18627
5-3-2-3-0	80.90	1440.00	6.86	158	90.97	1440.00	6.32	16064	93.25	1440.00	4.75	22581
5-3-3-3-0	81.62	1440.00	8.53	176	90.97	1440.00	7.82	14826	93.71	1440.00	4.71	16343
6-2-1-3-0	81.62	1440.00	5.81	408	86.88	1440.00	11.59	21680	89.70	1440.00	10.05	23709
6-2-2-3-0	80.99	1440.00	8.65	363	86.88	1440.00	11.06	23584	88.94	1440.00	10.83	19921
6-2-3-3-0	80.92	1440.00	9.45	306	86.93	1440.00	12.62	24913	89.01	1440.00	10.86	21820
6-3-1-3-0	83.45	1052.14	0.00	225	88.18	1440.00	7.28	18092	91.03	1440.00	3.46	9251
6-3-2-3-0	82.42	1440.00	7.61	250	88.19	1440.00	6.32	11083	91.43	1440.00	1.30	16389
6-3-3-3-0	82.42	1440.00	7.40	208	88.18	1440.00	7.16	14429	91.43	1440.00	5.12	7568
7-2-1-3-0	87.70	1440.00	6.67	551	83.26	1440.00	13.08	24620	89.33	1440.00	8.89	27864
7-2-2-3-0	84.99	1440.00	8.80	542	80.38	1440.00	16.75	28942	90.44	1440.00	8.16	25871
7-2-3-3-0	85.48	1440.00	9.22	551	80.38	1440.00	17.50	34500	90.05	1440.00	8.92	12784
7-3-1-3-0	78.61	1440.00	17.33	287	80.04	1440.00	18.20	12958	82.89	1440.00	14.72	8983
7-3-2-3-0	78.36	1440.00	17.10	245	80.03	1440.00	18.50	13523	82.89	1440.00	14.72	12081
7-3-3-3-0	78.40	1440.00	16.92	232	80.03	1440.00	18.37	8461	82.89	1440.00	14.72	11381
8-2-1-3-0	79.36	1440.00	17.01	552	79.53	1440.00	19.59	29577	89.14	1440.00	10.47	23569
8-2-2-3-0	76.67	1440.00	20.03	585	78.07	1440.00	21.72	28861	84.99	1440.00	13.35	20357
8-2-3-3-0	79.10	1440.00	17.22	471	80.64	1440.00	19.11	31196	89.43	1440.00	10.46	17962
8-3-1-3-0	83.70	1440.00	13.01	405	77.89	1440.00	20.72	18493	86.07	1440.00	11.71	15630
8-3-2-3-0	83.97	1440.00	12.97	408	77.92	1440.00	21.30	18474	85.37	1440.00	13.85	19862
8-3-3-3-0	83.69	1440.00	13.07	363	77.89	1440.00	21.14	21504	85.16	1440.00	14.12	27832
9-2-1-3-0	78.29	1440.00	17.66	513	76.21	1440.00	21.85	27881	83.28	1440.00	15.86	19936
9-2-2-3-0	76.86	1440.00	17.29	372	76.22	1440.00	21.72	21460	83.74	1440.00	15.59	7822
9-2-3-3-0	77.39	1440.00	19.41	368	75.94	1440.00	22.97	22559	83.14	1440.00	16.86	12675
9-3-1-3-0	80.70	1440.00	16.78	291	79.63	1440.00	19.26	18110	86.24	1440.00	13.76	14374
9-3-2-3-0	83.89	1440.00	13.44	263	83.67	1440.00	15.96	14780	89.14	1440.00	10.17	17554
9-3-3-3-0	82.42	1440.00	15.35	284	81.73	1440.00	17.67	13008	87.16	1440.00	12.64	21304
10-2-1-3-0	83.81	1440.00	14.08	395	80.22	1440.00	19.47	19066	84.50	1440.00	15.37	18228
10-2-2-3-0	82.38	1440.00	15.34	462	77.22	1440.00	22.61	19107	85.60	1440.00	14.39	12840
10-2-3-3-0	83.51	1440.00	14.38	312	79.96	1440.00	19.96	13727	84.76	1440.00	15.15	16469
10-3-1-3-0	86.00	1440.00	10.79	314	83.63	1440.00	14.12	7695	89.01	1440.00	10.26	13258
10-3-2-3-0	84.52	1440.00	13.31	329	80.34	1440.00	17.51	11064	89.09	1440.00	9.62	9969
10-3-3-3-0	63.69	1440.00	36.02	241	60.81	1440.00	38.16	6393	67.44	1440.00	31.73	8916
Average	81.42	1381.75	13.20	343	81.64	1440.00	16.13	19349	87.36	1440.00	11.12	16484

Tabela 2 – continuação

Com base nos resultados obtidos em (CRUZ; CUNHA, 2023), conclui-se que o modelo aprimorado com desigualdades válidas apresentou melhorias significativas nas

relaxações da Programação Linear. A otimização permitiu uma comparação direta entre as relaxações dos dois modelos, revelando que o algoritmo *Branch-and-Cut* aprimorado produziu relaxações, em média, 25% mais nítidas em comparação com o modelo básico. Esse resultado é especialmente notável em instâncias com até 40 pontos, no qual o algoritmo foi capaz de encontrar rapidamente soluções viáveis. No entanto, o desafio surgiu em instâncias maiores, em que o algoritmo teve dificuldade em encontrar soluções inteiras viáveis. Essa constatação destaca a necessidade contínua de pesquisa e desenvolvimento de abordagens que possam lidar eficientemente com problemas mais complexos e de maior escala (CRUZ; CUNHA, 2023).

O artigo Wolfinger e Salazar-González (2021) apresenta uma nova formulação de inteiro misto baseada em arco para o problema de coleta e entrega com cargas divididas. Os resultados demonstram que o PDPSLT pode fornecer soluções mais econômicas do que outras abordagens existentes, utilizando um modelo de algoritmo *Branch-and-cut* para realizar experimentos computacionais com objetivo avaliar sua eficácia. No entanto, para validar ainda mais os benefícios do PDPSLT e explorar seu potencial prático, é necessário resolver instâncias maiores do problema. Uma sugestão para pesquisas futuras é o desenvolvimento de formulações de particionamento de conjuntos para lidar de maneira mais eficiente com as restrições relacionadas às visitas a locais específicos pelo mesmo veículo. A sincronização do tempo nas localizações de transbordo também é identificada como um desafio importante.

Os dois trabalhos, Wolfinger e Salazar-González (2021) e Wolfinger e Salazar-González (2021), apresentados demonstram bons resultados para instâncias pequenas em relação ao modelo do problema apresentado, já para modelos do mundo real recomenda-se a obtenção de modelos práticos. Essa análise deve ser conduzida por meio de abordagens de solução meta-heurística capazes de lidar com instâncias de tamanho real.

3.1 Heurísticas para o Problema de Roteamento de Veículos

Nos últimos anos, houve o desenvolvimento de vários algoritmos sofisticados de decomposição de programação matemática para a resolução do Problema de Roteamento de Veículos (VRP). No entanto, apesar dos esforços empregados, apenas instâncias relativamente pequenas, podem ser resolvidas de forma otimizada, e o tempo de computação varia significativamente. No entanto, no contexto de ambientes da vida real, as instâncias do VRP frequentemente são grandes e exigem soluções rápidas e previsíveis. Isso implica na necessidade de heurísticas eficientes, na prática. Além disso, devido à variação na definição exata do problema em diferentes cenários, é essencial desenvolver heurísticas que sejam suficientemente flexíveis para lidar com uma variedade de objetivos e restrições secundárias. Essas preocupações têm sido claramente evidenciadas nos

algoritmos desenvolvidos nos últimos anos.

Heurísticas Construtivas são frequentemente utilizadas para gerar uma solução inicial para uma heurística de aprimoramento. No entanto, a maioria das heurísticas modernas é tão robusta que pode ser iniciada a partir de qualquer solução inicial, seja ela aleatória ou não, não havendo a necessidade de que seja uma solução viável (TOTH et al., 2015). No entanto, em Toth et al. (2015) , mesmo com a maioria heurísticas mencionadas, caiu em desuso, todavia, o trabalho apresentam dois métodos clássicos que ainda possuem um interesse particular, que são: a Heurística de Poupança de Clarke e Wright e o Algoritmos de Pétala.

A Heurística de Poupança de Clarke e Wright constrói inicialmente rotas de ida e volta e gradualmente as mesclam aplicando um critério de salvamento. Mais especificamente, mesclando as duas rotas em uma única rota, gerando uma economia. Uma vez que a poupança permanece a mesma ao longo do o algoritmo (TOTH et al., 2015).

O algoritmo de Pétala pretende gerar um conjunto S de rotas viáveis para o Problema de Roteamento de Veículos (VRP) e combiná-las por meio da solução de um problema de particionamento de conjuntos. Esses algoritmos abordam o desafio NP-difícil de minimizar o custo total das rotas a abordagem do VRP usando uma formulação de particionamento de conjuntos requer a resolução de um problema complexo. Normalmente, é empregada uma abordagem heurística para gerar um conjunto de rotas promissoras. Além disso, são geradas configurações conhecidas como "2-pétalas", que consistem em duas rotas que se sobrepõem ou se cruzam. Em Toth et al. (2015), os autores demostram 14 instâncias de referência, eles obtiveram soluções rapidamente com custos que estavam dentro de 2,38% das soluções mais conhecidas. O algoritmo Petal são especialmente adequados para lidar com problemas que possuem restrições adicionais, como janelas de tempo, além das restrições de capacidade e duração da rota. Nesses casos, a geração de colunas torna-se uma metodologia preferencial para selecionar soluções, especialmente em aplicações com restrições significativas.

Segundo Machado et al. (2021), uma heurística construtiva é um algoritmo que constrói iterativamente um solução no espaço do problema de acordo com uma regra pré-determinada e para quando está totalmente construída. Segundo o autor acredita-se que as heurísticas de construção encontram solução dentro de 10% a 15% da otimização e, portanto, podem ser usadas como uma solução inicial para uma melhoria de loop.

Ao longo da última década, classificar os métodos avançados para o Problema de Roteamento de Veículos com Capacidade (CVRP) tem se tornado mais fácil. No entanto, à medida que a pesquisa avança, tem surgido uma ampla variedade de métodos híbridos, que combinam conceitos de diferentes paradigmas algorítmicos (TOTH et al., 2015).

Esses métodos híbridos têm se baseado em ideias provenientes de diversas áreas, como busca local, busca de vizinhança Variada e busca baseado em população, programação inteira, programação por restrição, busca em árvore, mineração de dados e computação paralela. Com isso, a fronteira entre os métodos está se tornando cada vez

mais difusa (TOTH et al., 2015).

Em suma, os estudos abordam estratégias híbridas que se tornam objeto de discussão. Essas abordagens, ao amalgamar o que há de melhor em diversas técnicas algorítmicas, revelam-se capazes de proporcionar um desempenho aprimorado na resolução do CVRP.

Segundo o autor Wang e Lu (2009) uma abordagem híbrida frequentemente supera qualquer um dos métodos usados independentemente, existe grande quantidades de aplicações bem-sucedidas favorecem a utilização de abordagens híbridas.

As meta-heurísticas atuais para o Problema de Roteamento de Veículos com limitação de Carga (CVRP) podem ser amplamente classificadas em métodos de busca que combinam a exploração do espaço de soluções e a busca local mais elaborada. Muitas dessas meta-heurísticas são baseadas em comportamentos de um certo tipo de população. Os métodos de busca local exploram o espaço de soluções movendo-se a cada iteração de uma solução para outra solução em sua vizinhança (KIRKPATRICK; JR; VECCHI, 1983), enquanto que heurísticas baseadas em população são caracterizadas pela busca de uma população, na qual as soluções estabelecidas tendem a alcançar novos e diferentes espaços de busca de soluções, na esperança de gerar soluções melhores. Essa categoria, meta-heurística, engloba técnicas como otimização por colônia de formigas, algoritmos genéticos, busca de dispersão e religação de caminhos, dentre outras (TOTH et al., 2015).

Este estudo apresenta uma abordagem inovadora para o Problema de Roteamento de Veículos, utilizando um Algoritmo Híbrido de Sistema de Colônia de Formigas e Algoritmo de Otimização*Brainstorm* para criar roteiros de viagem multi-dia, abordando as limitações da abordagem tradicional do Problema do Caixeiro Viajante. O Algoritmo Híbrido de Sistema de Colônia de Formigas e Otimização por Tempestade de Cérebro supera algoritmos convencionais, como o Algoritmo Genético, Busca Tabu e Recozimento Simulado, em quatro dos cinco principais critérios: valor de aptidão, número de Pontos de Interesse incluídos no itinerário, avaliação média, custo total e duração total da viagem (HENDRAWAN; BAIZAL; WULANDARI, 2024).

O Algoritmo Híbrido de Sistema de Colônia de Formigas e Otimização por Tempestade de Cérebro alcança um valor médio de aptidão de 0,6704 em cinco conjuntos de 30 Pontos de Interesse aleatórios, superando o Algoritmo Genético (0,6337), Busca Tabu (0,6469) e Recozimento Simulado (0,6626). Ele também otimiza atributos de duração de viagem, custo e avaliação de forma mais eficaz. Por exemplo, gera um itinerário que requer apenas 68,3 horas (aproximadamente seis dias) para visitar 40 Pontos de Interesse, enquanto outros algoritmos precisam de pelo menos sete dias. Além disso, alcança valores de aptidão de 0,8628 e 0,8263 para os atributos de custo e avaliação, respectivamente, para 30 Pontos de Interesse (HENDRAWAN; BAIZAL; WULANDARI, 2024).

Na Busca local uma solução inicial pode ser aprimorada visando encontrar um mínimo local. Neste caso, um conjunto de modificações no atual solução é definida

para gerar uma vizinhança de soluções, na qual cada modificação é conhecida como um movimento. A melhor solução então se torna a nova solução atual no próximo loop de busca local, desde que o movimento melhore o custo. A busca local é repetida enquanto houver pelo menos um movimento que seja viável e melhore a atual solução, (MACHADO et al., 2021).

Busca Vizinha Variável (VNS, do inglês*Variable Neighborhood Search*), é uma metaheurística para resolução de problemas combinatórios que consiste em duas etapas principais: agitação e busca local. A ideia básica é uma mudança metódica de vizinhança combinando uma fase de descida para encontrar um ótimo local e uma fase de agitação para sair do vale correspondente (HADDADENE; LABADIE; PRODHON, 2016). Machado et al. (2021) apresenta os resultados dos benchmark utilizados para o benchmark do tipo A, em termos da melhor solução encontrada, Tabela 3. O algoritmo MGV-CP encontrou BKS (coluna AVG) em 21 das 27 ocorrências, revelando que pelo menos um BKS (melhor coluna) foi encontrado para cada instância. Os algoritmos propostos HGA-VNS e MGV-CP demonstraram resultados semelhantes e um desempenho melhor em comparação aos outros algoritmos apresentados. No entanto, o MGV-CP obteve melhores resultados para instâncias menores, enquanto o HGA-VNS geralmente encontrou os melhores resultados para instâncias maiores. No benchmark tipo B, a Tabela 3 demonstra que o MGV-CP encontrou BKS (coluna AVG) em 18 de 23 casos. Mais uma vez, os algoritmos HGA-VNS e MGV-CP reportaram resultados semelhantes e um desempenho superior em comparação aos outros algoritmos. O MGV-CP obteve os melhores resultados para conjuntos de instâncias pequenas e médias, bem como para as duas maiores instâncias. Por sua vez, o HGA-VNS produziu soluções ideais para as maiores instâncias (MACHADO et al., 2021).

Instância	DELS	LNS-ACO	OHGA	CVRP-FA		HGA-VNS		MGV-CP
			Best	Avg	Best	Avg	Best	Avg
A-n32-k5	784	784	784	796	798.1	784	802	784
A-n33-k5	661	661	661	661	661	679	661	661
A-n33-k6	742	742	742	742	742.7	742	760	742
A-n34-k5	778	778	778	778	778	779	796	778
A-n36-k5	799	799	799	799	804.2	799	799	799
A-n37-k5	669	669	669	669	669	669	669	669
A-n37-k6	949	949	949	949	955.5	949	949	949
A-n38-k5	730	730	730	730	730.7	730	730	730
A-n39-k5	822	822	822	822	822	822	822	822
A-n39-k6	831	831	833	831	834.6	831	831	831
A-n44-k6	937	937	937	937	937	937	937	937
A-n45-k6	944	944	958	953	953.4	944	944	945.2
A-n45-k7	1146	1146	1146	1147	1153.3	1146	1146	1146
A-n46-k7	914	914	914	914	914	919	931	914
A-n48-k7	1073	1084	1073	1073	1073	1090	1073	1073

Tabela 3 – Resultados comparativos para o conjunto A, adaptado de (MACHADO et al., 2021)

Instância	DELS	LNS-ACO	OHGA	CVRP-FA		HGA-VNS		MGV-CP
			Best	Avg	Best	Avg	Best	Avg
B-n31-k5	672	672	672	672	672	672	672	672
B-n34-k5	788	788	788	788	789.7	788	788	788
B-n35-k5	955	955	955	955	955.1	955	955	955
B-n38-k6	805	805	805	806	806.2	807	822	805
B-n39-k5	549	549	549	550	553.9	552	566	549
B-n41-k6	829	829	829	829	829.8	829	829	829
B-n43-k6	742	742	742	742	742	742	742	742
B-n44-k7	909	909	909	909	913.3	909	921	909
B-n45-k5	751	751	751	751	754.2	751	763	751
B-n45-k6	678	678	680	686	692.8	678	690	678
B-n50-k7 1	741	741	741	741	744.8	741	753	741
B-n50-k8	1313	1319	1315	1318	1329.6	1315	1324	1313
B-n51-k7	1033	1032	–	1032	-	1037	1053	1032
B-n52-k7	747	747	747	747	747.6	751	768	747
B-n56-k7	707	707	711	709	713.9	710	728	707
B-n57-k7	1153	–	1153	1162.5	1153	1153	1155	
B-n57-k9	1599	1598	1603	1610	1615.2	1598	1598	1598

Tabela 4 – Resultados comparativos para o conjunto B, adaptado de (MACHADO et al., 2021)

A busca Tabu (TS, do inglês *Tabu Search*) é meta-heurística que foi inicialmente desenvolvida por Glover F.;Kochenberger (2003). Ela propõe uma solução para problemas de programação inteira que foi formalizada e publicada em Glover F.;Kochenberger (2003) em uma série de trabalhos, abrangendo diversas aplicações. Ao longo da experiência, o autor constatou a eficiência da busca tabu na resolução de diferentes problemas, consolidando-a como uma técnica robusta. De forma geral, a busca tabu é um procedimento adaptativo de busca local, que possui uma estrutura de memória e é capaz de aceitar movimentos de piora quando não há possibilidade de melhoria. Essa abordagem permite explorar o espaço de busca de maneira mais abrangente, evitando ficar preso em mínimos locais e buscando soluções mais promissoras.

Em Obaid (2018) , o autor apresenta em seu trabalho os resultados dos experimentos realizados. Os resultados obtidos com o Algoritmo de Busca Tabu (TSA) demonstraram sua capacidade de expandir as soluções com qualidade para o CVRP de forma eficaz, mantendo um tempo de cálculo viável. O Problema de Roteamento de Veículos com limitação de Carga (CVRP) faz parte das classes de problemas não determinísticos de tempo polinomial (NP), o que impede a sua resolução em tempo polinomial, ou seja, a complexidade do problema cresce de forma polinomial em relação ao tamanho da entrada, na Tabela 5 o TSA mostrou ser especialmente eficiente em instâncias com tamanho pequeno, especialmente quando os veículos possuem capacidade média. Nesses casos, o tempo de execução foi reduzido, levando a resultados mais favoráveis (OBAID, 2018).

Algoritmos baseados em população são influenciados por conceitos naturais, como

Instâncias	Resultado Ótimo	N° Veículos	Sobrecarga	Tempo execução
H.M.S				
E-n101-k14	1122	14	1	00:04:26
E-n101-k8	835	8	3	00:02:54
E-n13-k4	247	4	0	00:00:06
E-n22-k4	375	4	0	00:00:11
E-n23-k3	569	3	0	00:00:09
E-n30-k3	506	4	0	00:00:14
E-n31-k7	427	7	0	00:00:28
E-n33-k4	835	4	0	00:00:17
E-n51-k5	521	5	0	00:00:35
E-n76-k10	830	10	0	00:01:59
E-n76-k14	1021	15	0	00:03:00
E-n76-k7	662	7	0	00:01:25
E-n76-k8	756	8	0	00:01:35

Tabela 5 – Resultados experimentais (OBAID, 2018)

.

a evolução das espécies e o comportamento de busca de alimentos de insetos sociais. Esses métodos implementam uma estratégia de alto nível que se baseia em diferentes estruturas de memória, como redes neurais, conjuntos de soluções representadas como cromossomos ou matrizes de feromônios. Além disso, todos os métodos conhecidos desse tipo para a heurística são utilizados para problemas de VRP e também incorporados em componentes de busca local para direcionar a busca em direção a soluções promissoras. Portanto, a maioria dos métodos baseados em população na literatura VRP apresenta um melhor resultado quando utilizado de forma híbrida (TOTH et al., 2015).

O algoritmo de otimização de colônia de formigas (ACO, do inglês *Ant Colony Optimization*), segundo Jia, Mei e Zhang (2021), é um processo geral do ACO inicia-se com os componentes e os parâmetros inicializados no algoritmo, então, em cada geração $n = |I| + 1$ as formigas constroem suas soluções uma a uma. As soluções geradas são avaliadas para atualizar a melhor solução global e a melhor solução de iteração, os limites dos feromônios são dados pela matriz de feromônio que atualizado a cada iteração ou trilha da formiga, iteração continua até que o critério de parada seja atendido ou o número de iteração definido, seja alcançado, por fim, todo o algoritmo termina e retorna à melhor solução global.

A construção de rotas será realizada por uma matriz de feromônios mantida para guiar as formigas na construção das rotas, para isso, utiliza-se a versão da ACO que define a função min-max (max–min *ant system-MMAS*). O tamanho da matriz de feromônios onde φmin e φmax define os valores de feromônio usados para manter a capacidade de exploração do algoritmo, $\varphi n \times n$, o qual $n = |I| + 1$ representa o número de depósitos e clientes onde r é inicializado vazio e I conjunto contendo o depósito e os clientes onde cada elemento de φ_{ij} define o valor de feromônio para formar a rota de i

para j. Para tanto, o algoritmo utiliza dois limites: φ min e φmax, valores usados para controlar a capacidade da exploração das formigas (JIA; MEI; ZHANG, 2021).

$$\varphi max = \frac{1}{(1-\rho)f(x^{g_b})}, \tag{6}$$

$$\varphi min = \frac{\varphi max(1-n\sqrt{p_r})}{(n/2-1)n\sqrt{p_r}}, \tag{7}$$

Sendo ρ o parâmetro que define a evaporação do feromônio, enquanto x^{g_b} representa a solução global, e $\sqrt{p_r}$ o valor de mutação normalmente definido em valor baixo de 0,05. Para geral solução inicial representada por x^{g_b}, o método de inserção é aplicado para gerar uma *tour* gigante que posteriormente é divida em rotas viáveis, e após gerar todas as rotas, o algorítimo aplica um método de busca local para a rota a fim de otimizá-la (JIA; MEI; ZHANG, 2021).

Otimização de colônia de formigas de dois níveis (BACO, do inglês *Bilevel ant colony optmization*), segundo Jia, Mei e Zhang (2021), apresenta os resultados obtidos separando em duas tabela, a Tabela 6 comparando o algoritmo utilizando sete instancias de pequena escala e na Tabela 7 dez instâncias de grande escala, foi realizado comparação do BACO com cinco algoritmos, VNS, simulado recozimento (SA, do inglês *Simulated Annealing*), Algorítimo Genético (AG),Busca Local Iterativa (ILS, do inglês *Iterated Local Search*) e um Sistema formiga maximização e minimização (MMAS, do inglês *Max-Min Ant System*) , o SLMMAS e um ILS com MMAS de nível único, também foi realizado com comparação o algoritmo ILS, esse foi originalmente proposto para resolver o (EVRP, do inglês*Electric Capacitated Vehicle Routing Problem*) com carregamento não linear, mas pode ser facilmente usado para resolver o CVRP.

Com base nos resultados dos autores Jia, Mei e Zhang (2021), pode-se concluir que embora o ACO tenha apresentado desempenho promissor, existem questões em aberto a serem analisadas, os resultados experimentais mostram que a capacidade de BACO em resolver as instâncias grandes é fraco em relação à resolução de instâncias regulares.

No algoritmo genético híbrido (AGH) o autor explana como sendo um algorítimo para explorar globalmente as soluções, enquanto as heurísticas se concentram na exploração de soluções locais. Essa abordagem híbrida pode ser aplicada de várias maneiras, incorporando a heurísticas na inicialização da estrutura da população podendo utilizar elitismo ou incluir também uma busca local no processo de reprodução AG, para melhora a descendência. Em aplicações práticas, a forma da abordagem híbrida depende fortemente do tipo de problema otimizado, (WANG; LU, 2009).

Seguindo essa abordagem de AGH Wang e Lu (2009) apresenta os resultados dos seus experimentos na Tabela 8. Nesta tabela são apresentadas a porcentagem de desvio entre a solução próxima até o ótimo encontrado e o valor mais conhecido foi derivado para avaliar a capacidade de solução do AGH proposto. Ainda na Tabela 8 e possível ver o resumo dos cálculos das rotas otimizada para o problema.

O autor (LI; GOLDEN; WASIL, 2005) propõe um algoritmo de "record-to-record travel" (RTR), otimizado com uma lista de vizinhos de comprimento variável, para

Case	Index	BKS	SLMMAS	ILS	GA	SA	VNS	BACO
	min	384.67	385.44	**384.67**	**384.67**	**384.67**	**384.67**	**384.67**
E22	mean		385.44↑	385.69↑	**384.67**↕	**384.67**↕	**384.67**↕	**384.67**
	std		0.00	2.11	0.00	0.00	0.00	0.00
	min	571.94	582.61	590.35	**571.94**	**571.94**	**571.94**	**571.94**
E23	mean		582.94↑	592.05↑	**571.94**↕	**571.94**↕	**571.94**↕	**571.94**
	std		0.60	1.04	0.00	0.00	0.00	0.00
	min	509.47	514.42	**509.47**	**509.47**	**509.47**	**509.47**	**509.47**
E30	mean		516.11↑	**509.47**↕	**509.47**↕	**509.47**↕	**509.47**↕	**509.47**
	std		0.61	0.00	0.00	0.00	0.00	0.00
	min	840.14	859.25	840.57	844.25	840.57	**840.14**	840.57
E33	mean		860.08↑	844.07↑	845.62↑	854.07↑	**840.43**↓	842.30
	std		0.74	7.78	0.92	12.80	1.18	1.42
	min	529.90	549.81	**529.90**	**529.90**	533.66	**529.90**	**529.90**
E51	mean		564.93↑	539.03↑	542.08↑	533.66↑	543.26↑	**529.90**
	std		8.70	6.92	8.57	0.00	3.52	0.00
	min	692.64	724.24	694.64	697.27	701.03	**692.64**	**692.64**
E76	mean		762.09↑	704.24↑	717.30↑	712.17↑	697.89↑	**692.85**
	std		13.23	7.15	9.58	5.78	3.09	0.81
	min	839.29	891.77	841.02	852.69	845.84	**839.29**	840.25
E101	mean		904.62↑	851.62↑	872.69↑	852.48↑	853.34↑	**845.95**
	std		4.64	6.93	9.58	3.44	4.73	4.58
w/t/l	-	-	7/0/0	6/1/0	4/3/0	4/3/0	3/3/1	

Tabela 6 – Comparação entre o BACO e os algoritmos, instâncias em pequena escala (JIA; MEI; ZHANG, 2021)

Case	Index	BKS	SLMMAS	ILS	GA	SA	VNS	BACO
	min	16028.05	17238.76	16058.29	16488.60	16610.37	16028.05	**15901.23**
X143	mean		17985.91↑	16318.57↑	16911.50↑	17188.90↑	16459.31↑	**16031.46**
	std		415.20	160.07	282.30	170.44	242.59	262.47
	min	11674.96	11421.43	11323.56	11762.07	11404.44	11323.56	**11133.14**
X214	mean		12023.72↑	11537.58↑	12007.06↑	11680.35↑	11482.20↑	**11219.70**
	std		129.22	72.55	156.69	116.47	76.14	46.25
	min	27064.88	31927.71	27947.89	28008.09	27222.96	27064.88	**26478.34**
X352	mean		31927.71↑	28364.41↑	28336.07↑	27498.03↑	27217.77↑	**26593.18**
	std		0.00	142.04	205.29	155.62	86.20	72.86
	min	25370.80	28448.27	26511.28	26048.21	27222.96	25370.80	**24763.93**
X459	mean		29434.48↑	26726.69↑	26345.12↑	25809.47↑	25582.27↑	**24916.60**
	std		540.16	126.93	185.14	157.97	106.89	94.08
	min	51929.24	56609.94	53102.46	54189.62	**51929.24**	52181.51	53822.87
X573	mean		57170.35↑	53507.46↓	55327.62↑	52793.66↓	**52548.09**↓	54567.15
	std		199.81	275.76	548.05	577.24	278.85	231.05
	min	71345.40	84435.72	74409.65	73925.56	72549.90	71345.40	**70834.88**
X685	mean		84435.72↑	75087.58↑	74508.03↑	73124.98↑	71770.57↑	**71440.57**
	std		0.00	259.60	409.43	320.07	197.08	281.78
	min	81002.01	90441.6	84298.43	84034.73	81392.78	81002.01	**80299.76**
X749	mean		90441.36↑	84860.28↑	84759.79↑	81848.13↑	81327.39↑	**80694.54**
	std		0.00	287.26	376.10	275.26	176.19	223.91
	min	164289.95	171553.21	168651.19	170965.68	165069.77	**164289.95**	164720.80
X819	mean		172355.34↑	169837.06↑	172410.12↑	165895.78↑	**164926.41**↓	165565.79
	std		311.09	483.35	568.58	403.70	318.62	401.02
	min	341649.91	353046.07	348733.86	357391.57	342796.88	**341649.91**	342993.01
X916	mean		354262.15↑	350822.41↑	360269.94↑	343533.85↓	**342460.70**↓	344999.95
	std		464.68	1177.08	229.19	556.98	510.66	905.72
	min	77476.36	89611.93	79493.37	78832.90	78053.86	77476.36	**76297.09**
X1001	mean		89611.93↑	79928.29↑	79163.34↑	NA↑	77920.52↑	**77434.33**
	std		0.00	265.91	NA	306.27	234.73	719.8671
w/t/l	-	-	10/0/0	9/0/1	10/0/0	8/0/2	7/0/3	-

Tabela 7 – comparação entre o BACO e os algoritmos, instâncias em grande escala (JIA; MEI; ZHANG, 2021)

resolver problemas de roteamento de veículos (VRP) em grande escala. O algoritmo se destacou por sua velocidade e precisão ao solucionar 20 problemas “large-scale vehicle

Problema Nº	Cruzamento probabilidade	Mutação probabilidade	CV para CVRP solução	CVRP %	Melhor solução conhecida solução	deviação valor
E-n23-k3	1.00	0.74	0.32	568.56	568.563	0.00
E-n30-k3	0.98	0.75	0.00	508.14	538.958	5.72
E-n33-k4	1.00	0.85	0.85	845.24	838.721	0.78
E-n51-k5	0.65	0.88	0.41	524.61	524.944	0.06
E-n76-k7	0.74	0.70	0.84	701.28	687.603	1.99
E-n76-k8	0.62	0.73	0.73	750.48	735	2.11
E-n76-k10	1.00	0.85	0.73	853.05	832	2.53
E-n76-k14	1.00	0.76	0.67	1057.70	1032	2.49
E-n101-k8	0.88	0.77	1.13	847.50	817	3.73
E-n101-k14	1.00	0.81	0.66	1121.30	1077	4.11

Tabela 8 – Resultados AGH (WANG; LU, 2009)

routing problems“ (LSVRP). Além disso, os autores geraram um novo conjunto de 12 VRPs de grande escala, contendo entre 560 e 1200 clientes, para testar a eficácia do algoritmo. Os resultados mostraram que o algoritmo foi capaz de produzir soluções rápidas e precisas para esses problemas complexos. O estudo também aplicou o algoritmo “ variable-length neighbor list record-to-record travel solution “ (VRTR) a sete problemas benchmark de pequena escala, demonstrando que ele gera soluções de alta qualidade em um tempo de computação reduzido, competindo eficientemente com o procedimento de busca tabu granular “ granular tabu search solution “ (GTS).

Problema	n	Melhor conhecido	Fonte	Porcentagem da melhor solução conhecida							
				EST	$\alpha = 1$	$\alpha = 0.6$	$\alpha = 0.4$	EST	$\alpha = 1$	$\alpha = 0.6$	$\alpha = 0.4$
21	560	16212.83	EST	16212.83	16602.99	16627.22	16739.25	0.00	2.41	2.56	3.25
22	600	14641.64	ORTR	14652.28	14651.27	14655.60	14669.39	0.07	0.07	0.10	0.19
23	640	18801.13	EST	18801.13	19005.37	19094.98	18838.62	0.00	1.09	1.56	0.20
24	720	21389.43	EST	21389.43	21784.43	21616.25	21932.57	0.00	1.85	1.06	2.54
25	760	17053.26	EST	17053.26	17151.43	17163.31	17146.41	0.00	0.58	0.65	0.55
26	800	23977.74	EST	23977.74	24189.66	24200.27	24009.74	0.00	0.88	0.93	0.13
27	840	17651.60	ORTR	18253.55	17823.40	17936.25	17901.56	3.41	0.97	1.61	1.42
28	880	26566.04	EST	26566.04	26606.11	26784.38	26787.38	0.00	0.15	0.82	0.83
29	960	29154.34	EST	29154.34	29181.21	29183.58	29401.90	0.00	0.09	0.10	0.85
30	1040	31742.64	EST	31742.64	31976.73	31961.58	33246.60	0.00	0.74	0.69	4.74
31	1120	34330.94	EST	34330.94	35369.17	35355.50	36339.65	0.00	3.02	2.98	5.85
32	1200	36919.24	EST	36919.24	37421.44	37410.84	39415.85	0.00	1.36	1.33	6.76
Percentual médio da melhor solução conhecida.								**0.29**	**1.10**	**1.20**	**2.28**
Tempo médio de computação (min)									**3.16**	**2.94**	**2.08**

Tabela 9 – Resultados do VRTR com diferentes valores do parâmetro α.

No Aprendizado por Reforço Profundo *(*do inglês Deep Reinforcement Learning), um dos primeiros modelos de aprendizagem profundo (*Deep Learning*) utilizado para o problema de roteamento foi o "*Pointer Network*", que usou aprendizado supervisionado para resolver o TSP (VINYALS; FORTUNATO; JAITLY, 2015) e, posteriormente, foi estendido para o aprendizado por reforço (*Reinforcement Learning-RL*) Bello et al. (2017). Depois, o "*Pointer Network*" foi adotado para resolver o CVRP Nazari et al. (2018), no qual a arquitetura de rede neural de recorrência no codificador era removida para reduzir a complexidade de computação sem degradar a qualidade da solução. Para melhorar ainda mais o desempenho, uma arquitetura baseada em transformador foi incorporada, integrando autoaprendizado tanto no codificador quanto no decodificador Kool, Hoof e Welling (2019). Diferente de outros métodos, que aprendem por heurísticas, O*NeuRewriter* foi proposto para aprender como escolher a próxima solução em um *framework* de busca local (NAZARI et al., 2018). Apesar de seus resultados serem promissores, esses métodos são menos eficazes para solucionar o CVRP. Recentemente, alguns métodos baseados em aprendizagem têm sido propostos para resolver CVRP, sendo estes inspirados por abordagem de RL multiagente (NAZARI et al., 2018).

Em Jia, Mei e Zhang (2021) realizou-se uma primeira tentativa de resolver a soma mínima de um CVRP por meio de cooperativas ações de múltiplos agentes para construção de rotas. Já em Nazari et al. (2018) propôs-se um controlador baseado em RL para selecionar entre várias meta-heurísticas com características diferentes para resolver a soma mínima CVRP. Embora tenha produzindo melhor desempenho do que as heurísticas convencionais, a abordagem proposta é incapaz de lidar bem visando mínimo-máximo necessários para os veículos.

Já em Li et al. (2022), o autor utilizou aprendizado por reforço profundo (DRL, do inglês *Deep reinforcement learning*) no problema do Roteamento de Veículos com limitação de carga com critérios de min-max e soma mínima para alcançar os objetivos. Foi proposto, também, um método heurístico construtivo que utiliza DRL e um mecanismo de atenção para aprender uma política para a construção de rotas. Especificamente, a rede de políticas consiste em um codificador, um decodificador para seleção de veículo e um decodificador para seleção de nó. Esses componentes trabalham em conjunto para escolher um veículo e um nó para esse veículo em cada etapa do processo de construção de rotas.

Os resultados experimentais demonstram que o desempenho geral do presente método apresentado supera a maioria das heurísticas convencionais e é competitivo em relação aos métodos heurísticos de última geração, como o SISR, porém, com um tempo de computação significativamente menor. Além disso, quando comparado a outro método baseado em aprendizado, o método do autor apresenta melhores resultados, mesmo com tempos de computação comparáveis (LI et al., 2022). Ademais, o método proposto possui boa capacidade de generalização para problemas com mais clientes, mostrando-se robusto e eficiente em cenários mais complexos. Na Tabela 10, observa-se

que o método DRL tende a ter um desempenho melhor do que o VNS em instâncias distribuídas uniformemente e é ligeiramente inferior nas instâncias de distribuição não uniforme para ambos, o método DRL é capaz de gerar soluções de qualidade de solução comparável com tempo de computação muito menor (LI et al., 2022).

Assim, o autor (LI; GOLDEN; WASIL, 2005) concluiu que a técnica VRTR foi aplicada a sete problemas de roteamento de veículos (VRPs) de pequena escala, sem a necessidade de ajustes adicionais. Os resultados obtidos foram comparados com outra técnica de otimização conhecida, mostrando que a VRTR gerou soluções de alta qualidade, com desempenho próximo das melhores soluções conhecidas e em um curto tempo de processamento. As diferentes versões da VRTR demonstraram ser rápidas, precisas e altamente competitivas em termos de eficiência e qualidade das soluções geradas.

Distância	Instância	Resultado Ótimo	DLT	VNS	
Min-Max	Uniforme	P-n60-k10	-	306	308
Min-Max	Uniforme	A-n61-k9	-	319	307
Min-Max	Uniforme	E-n76-k7	-	372	375
Min-Max	Uniforme	A-n80-k10	-	795	813
Min-Max	Uniforme	E-n101-k8	-	446	455
Min-Max	Uniforme	Avg. Gap	-	4.11%	4.49%
Min-Max	Não Uniforme	B-n41-k6	-	385	371
Min-Max	Não Uniforme	B-n51-k7	-	392	378
Min-Max	Não Uniforme	B-n63-k10	-	564	558
Min-Max	Não Uniforme	M-n101-k10	-	419	401
Min-Max	Não Uniforme	CMT11	-	878	869
Min-Max	Não Uniforme	Avg. Gap	-	5.48%	2.59%

Tabela 10 – Resultados do método DLT e VNS para diferentes instâncias com critério de distância Min-Max. adaptado (LI et al., 2022)

Capítulo 4

Abordagem Híbrida de Metaheurística para CVRP e Resultados

O objetivo deste estudo é encontrar uma solução ótima (ou próxima) para o problema do CVRP. Para isso, foram implementadas e analisadas várias estratégias em um algoritmo computacional. A ideia é analisar soluções hibridas que possam otimizar os resultados do CVRP, minimizando seu custo operacional em um tempo viável de processamento.

Para realizar a validação serão aplicados os algoritmos em diversas instâncias de um dataset conhecido, a fim de comparar os valores gerados pelos algoritmos com os resultados já encontrados e documentados na literatura por pesquisadores que utilizaram os mesmos datasets, cujos trabalhos foram mencionados na seção 3.

No dataset *Capacitated Vehicle Routing Problem Library* (CVRPLIB)[1], foram utilizadas as instâncias das classes "A" e "B". A classe "A" contém localizações e demandas aleatórias dos clientes, enquanto a classe "B" apresenta instâncias agrupadas. Foram também utilizadas instâncias de grande escala, baseadas em um conjunto de dados de *benchmark*, pertencentes à classe "Li", conforme apresentado por (LI; GOLDEN; WASIL, 2005).

4.1 Algoritmos

O algoritmo descrito nesta seção foi implementado em Python 3, e todos os experimentos foram realizados em um notebook equipado com um processador Intel Core i5 de 8ª geração e 12 GB de RAM. Para avaliar a eficácia do nosso algoritmo proposto

[1] Disponível em <http://vrp.galgos.inf.puc-rio.br/index.php/en/.>

para o CVRP, foram aplicados conjuntos de benchmarks da CVRPLIB. Para esse fim, os problemas de benchmark foram obtidos do repositório CVRPLIB[2].

O objetivo desta pesquisa é desenvolver um algoritmo que aborde o Problema de Roteamento de Veículos Capacitados (CVRP). Para alcançar isso, várias estratégias serão implementadas e analisadas dentro de um algoritmo computacional. O objetivo é desenvolver soluções híbridas que possam otimizar os resultados do CVRP, minimizando os custos operacionais e garantindo tempos de processamento viáveis.

A justificativa para a adoção do método de Otimização por Colônia de Formigas (ACO) baseia-se nos resultados obtidos por (JIA; MEI; ZHANG, 2021), que demonstraram sua eficácia em encontrar soluções de alta qualidade em comparação com outros algoritmos, especialmente para grandes instâncias. Essa escolha está alinhada com o principal objetivo desta pesquisa: resolver o CVRP de forma eficiente dentro de um tempo computacional aceitável.

Uma estratégia envolve o uso de várias técnicas de metaheurísticas, incluindo Algoritmos Genéticos (AG), para gerar um conjunto inicial de soluções globais, que podem então ser refinadas usando Otimização por Colônia de Formigas (ACO). Para a fase de busca local, a Busca Tabu é utilizada devido à sua eficácia em explorar minuciosamente o espaço de soluções. Este método ajuda a evitar ótimos locais e a descobrir soluções mais promissoras (OBAID, 2018). Essa abordagem híbrida oferece benefícios tanto em termos de desempenho computacional quanto de qualidade das soluções (TOTH et al., 2015). De acordo com os resultados experimentais apresentados por (OBAID, 2018), o uso exclusivo da Busca Tabu resultou em bons tempos de execução para instâncias com até 75 rotas, onde cada veículo completou uma rota. O maior tempo foi observado para a instância com 101 rotas para clientes, onde 14 veículos percorreram as rotas em 4 minutos e 26 segundos, enquanto o menor tempo foi de 11 segundos para uma rota com 22 clientes usando 4 veículos, conforme detalhado na Tabela 5 na sessão 3.1

Assim, como apresentado nesta seção, a escolha do método híbrido que combina Algoritmo Genético com Otimização por Colônia de Formigas e Busca Tabu deve permitir uma exploração abrangente do espaço de busca, evitando mínimos locais e buscando soluções mais promissoras. Esta abordagem híbrida visa fornecer soluções eficientes e de alta qualidade para o CVRP ao AG+ACO e Busca Tabu, o que pode melhorar o desempenho e a eficácia na resolução de problemas do CVRP (TOTH et al., 2015).

No método híbrido proposto, as soluções são encontradas através de um processo iterativo. Em cada iteração, várias formigas podem encontrar soluções que "visitem"todos os nós. A quantidade de feromônio em cada aresta é atualizada com base nas soluções encontradas. A formiga que percorreu a menor distância é considerada a melhor solução local. Para garantir que a solução seja a mais otimizada, a Busca Tabu é aplicada. Se a solução local tiver uma distância menor do que a melhor de qualquer iteração anterior,

[2] Disponível em <http://vrp.galgos.inf.puc-rio.br/index.php/en/.>

ela se torna a melhor solução global. Formigas elites então depositam feromônios ao longo do caminho da melhor solução global para fortalecê-la ainda mais.

A atualização offline da trilha ocorre quando um conjunto de formigas gera rotas. No final de cada iteração, a evaporação e o reforço da trilha são realizados. Um método de atualização de feromônio baseado em classificação é usado, onde formigas com as melhores rotas recebem valores mais altos. Neste modelo, cada formiga pode ter um valor diferente na tabela de feromônio. Além disso, uma solução elitista pode ser implementada, onde o reforço é aplicado à melhor solução o tempo todo. Também é possível considerar penalizar a pior solução atribuindo-lhe um valor baixo na tabela de feromônio, tornando-a uma opção menos viável na tomada de decisões. Esta matriz desempenha um papel na orientação do processo de busca. Inicialmente, a matriz de feromônio é configurada com os valores de solução gerados pelo Algoritmo Genético (AG), representando a quantidade inicial de feromônio em cada aresta. À medida que as formigas percorrem o espaço de soluções, elas depositam feromônios com base na qualidade das rotas que encontram. A matriz é atualizada em duas fases principais: Evaporação: No final de cada iteração, os níveis de feromônio em todas as arestas são reduzidos de acordo com uma taxa de evaporação. Este processo simula o decaimento natural do feromônio ao longo do tempo, garantindo que caminhos mais antigos e menos favoráveis recebam menos atenção. Reforço: Após a evaporação, a matriz é atualizada para reforçar os caminhos percorridos pelas formigas que encontraram as melhores soluções. Formigas que encontram soluções mais curtas ou melhores depositam feromônios adicionais em seus respectivos caminhos, tornando essas rotas mais atraentes para futuras formigas. Esse reforço é feito usando um método baseado em classificação, onde formigas com rotas superiores contribuem com mais feromônio, enquanto rotas menos eficazes recebem pouco ou nenhum reforço. Como mostrado no fluxograma da Figura 2.

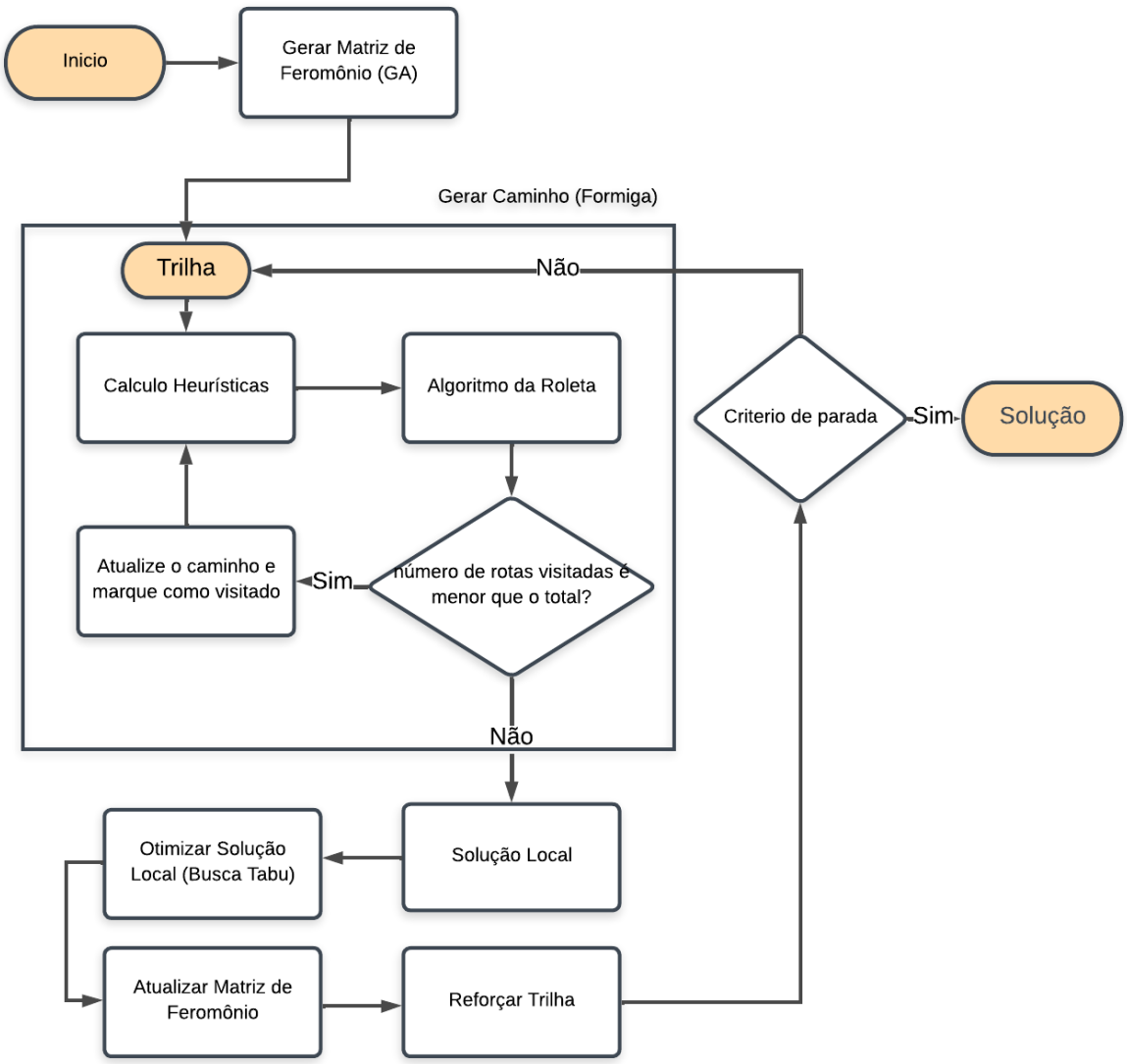

Figura 2 – Fluxograma do Algoritmo AG+ACO+TABU

O modelo de implementação pode ser visto na Figura 2 e no pseudocódigo abaixo, intitulado Algoritmo de Colônia de Formigas.

- Iniciar: Início do algoritmo.
- Gerar Matriz de Feromônio (AG): Inicialização da matriz que representa a trilha de feromônio.
- Gerar Caminho (Formiga): Cada formiga constrói uma solução parcial com base na matriz de feromônio e heurísticas locais.
- Solução Local: Avaliação da solução construída pela formiga.
- Otimizar Solução Local (Busca Tabu): Melhoria da solução usando técnicas como busca tabu para explorar o espaço de soluções.
- Atualizar Matriz de Feromônio: Atualizar a matriz de feromônio com base nas soluções encontradas pelas formigas.

- ❏ Reforçar Trilha: Reforçar as trilhas de feromônio para favorecer caminhos promissores.
- ❏ Condicional (Critérios de Parada): Verificar se os critérios de parada foram atendidos.
- ❏ Sim: A solução foi encontrada; finalizar o algoritmo. Não: Atualizar a matriz de feromônio e reiniciar o fluxo, retornando à etapa de geração de caminho (Gerar Caminho (Formiga)).

Algorithm 1 Algoritmo de Otimização por Colônia de Formigas

```
Input: iterações; formigas; evaporação; k; pior=False; elitista=False;
Output: melhor_custo; melhor_rota;
n = numero_de_clientes;
  trilha = criar matriz n × n preenchida com geração AG;
  trilha= AG(); melhor_rota = Nenhum;
  melhor_custo = [];
for i de 0 até iterações do
    lista = criar lista vazia;
    if forçar_parada(melhoresSoluções) é verdadeiro then
      parar o loop;
    end
    for a de 0 até formigas do
      solução = executar_formiga(trilha);
        custo = calcular_custo_rota(solução);
    end
    custo, solução = busca_tabu(1, k, 20, 1.05, custo, solução);
     adicionar(custo, solução) à lista;
     melhor_estático = custo;

    if custo < melhor_custo then
      melhor_custo = custo;
        melhor_rota = copiar solução;

    end
    adicionar melhor_estático à listaMelhoresSoluções;
    trilha = (1 - evaporação) × trilha;
    if pior é verdadeiro then
      custo, solução = encontrar_pior_solução(lista);
        reforçar(solução, -1, trilha);

    end
    if elitista é verdadeiro then
      reforçar(melhor_rota, 1, trilha);

    end
    ordenar lista por custo;
     delta = k;
     for custo, solução em lista[:k  ] do
      reforçar(solução, delta, trilha);
        delta -= 1

    end
end
return melhor_custo melhor_rota;
```

Na função *executar_formiga(trilha)*, onde a formiga seleciona a rota, é utilizada uma abordagem de minimização, onde a rota é escolhida com base no menor custo. Em cada iteração do algoritmo, as formigas são geradas para construir caminhos através de um grafo. Cada formiga seleciona o próximo nó a visitar com base na quantidade de feromônio na trilha e na capacidade do veículo. Uma heurística de minimização é aplicada que

considera os custos e demandas dos nós. À medida que as formigas constroem caminhos, elas atualizam a trilha de feromônio com base nas informações coletadas. Após um número de iterações, o algoritmo retorna a melhor solução encontrada. A continuação do Algoritmo 1 demonstra as variáveis usadas na função *executar_formiga(trilha)*:

Function executar formiga(trilha)

```
Input: trilha
Output: solução
n = número_de_rotas;
d = matriz_de_distâncias;
q = capacidade_do_veículo;
c = matriz_de_custos;
solucao = lista vazia;
maxc = valor maximo em c;
visitado = criar lista boolean com n elementos, inicializada como Falso;
contagem = 1;
while contagem < n do
    caminho =[0];
    v =0;
    carga = 0;
    while Verdadeiro do
        pode = lista de índices i onde visitado[i] é Falso e carga + d[i] ≤ q e v ≠ i;
        if tamanho de pode é 0 then
            parar o loop;
        end
        peso = criar lista com valores;
        max(trilha[v, i], min_trilha) para cada i em pode; # heurística de minimização, prioriza a rota mais
          curta
        heu = criar lista com valores (max c − c[v, i]) max c para cada i em pode;
        if v ≠ 0 then
            # se a carga for menor que 50%, priorizar rotas mais distantes do depósito
            if carga < q × 0.5 then
                heu = multiplique cada elemento de heu por 2
            end
            else
                heu = multiplique cada elemento de heu por 2 se c[0, i] < c[0, v], caso contrário,
                  multiplique por 1 para cada i em pode;
            end
        end
        heu = divida cada elemento de heu por max(heu);
        peso = divida cada elemento de peso por max(peso); peso = multiplique peso por heu;
        # algoritmo de roleta v = escolha aleatória de um elemento de pode com pesos peso;
        if v = 0 then
            parar o loop;
        end
        else
            adicione v ao caminho;
            carga+ = d[v];
            visitado[v] = Verdadeiro;
            contagem+ = 1;
        end
    end
    adicione caminho à solucao;
end
return solucao;
```

O algoritmo de Otimização por Colônia de Formigas possui condições de parada; essa função é responsável por analisar as soluções mais recentes e verificar se são idênticas. Se isso ocorrer, o algoritmo é interrompido, indicando que as formigas estão

convergindo para uma solução estável, evitando execução desnecessária quando não há mais melhorias possíveis.

Para validar a funcionalidade dos algoritmos propostos, testes foram realizados usando várias instâncias de referência obtidas do conjunto de dados CVRPLIB. Os resultados foram então comparados com descobertas registradas na literatura. No final da execução, a expectativa era que o algoritmo de otimização apresentasse as rotas mais conhecidas ou aquelas mais próximas do melhor; e também propusesse rotas otimizadas que atendam às restrições do CVRP.

À luz do contexto acima, esta pesquisa visa desenvolver um algoritmo baseado em inteligência artificial que utiliza uma solução metaheurística de Algoritmos Genéticos (AG), Otimização por Colônia de Formigas (ACO) e Busca Tabu (TS), para encontrar uma solução ótima - ou próxima do ótimo - para o CVRP.

Para alcançar o objetivo proposto, os seguintes passos foram estabelecidos:

- ❏ Desenvolver o algoritmo AG+ACO+TS para o CVRP.
- ❏ Realizar experimentos computacionais utilizando a instância de teste.
- ❏ Comparar os resultados obtidos com o algoritmo proposto com soluções ótimas conhecidas ou aquelas encontradas por outros métodos de otimização existentes.
- ❏ Analisar os resultados, identificando as vantagens, limitações e possíveis melhorias do algoritmo proposto.

Portanto, os passos mencionados são descritos na seção de resultados. Com base nessa análise, podemos afirmar que cada fase oferece uma visão abrangente e detalhada. Isso não só garante transparência nas operações, mas também serve como uma base sólida para revisões e melhorias futuras.

4.2 Resultados

4.3 Comparação dos Resultados

Esta seção apresenta uma análise comparativa dos resultados obtidos com o uso do algoritmo híbrido que combina Otimização por Colônia de Formigas (ACO) e Busca Tabu, com a incorporação de Algoritmos Genéticos (GA) para otimizar os custos no Problema de Roteamento de Veículos com Capacidade (CVRP).

Inicialmente, o algoritmo ACO com Busca Tabu foi testado. Embora esse método seja amplamente utilizado e geralmente eficaz, ele não convergiu para uma solução satisfatória antes do término das iterações. Apesar de utilizar os mesmos parâmetros e número de iterações do algoritmo híbrido proposto, os resultados não foram considerados aceitáveis para pequenas instâncias, pois o algoritmo falhou em encontrar soluções que

atendessem aos critérios de qualidade definidos. Acredita-se que a limitação ocorreu devido à tendência do ACO de se prender a ótimos locais, uma questão parcialmente mitigada pela Busca Tabu, mas ainda insuficiente para alcançar as melhores soluções em algumas instâncias.

Para ilustrar os resultados, foram utilizadas instâncias nomeadas de acordo com o tipo, número de clientes e número de veículos, como, por exemplo, E-n22-k4, onde "E"representa o tipo da instância, "22"indica o número de clientes e "4"o número de veículos utilizados. A instância E-n22-k4, por exemplo, apresentou um custo de 375,0 e um tempo de execução de 2,12 segundos, enquanto a instância B-n56-k7, com 56 clientes e 7 veículos, apresentou um custo de 707,0 e um tempo de execução de 91,49 segundos. Esse aumento no tempo de execução, à medida que o número de clientes e veículos cresce, reflete a complexidade computacional crescente do problema.

Os tempos de execução variaram substancialmente entre as instâncias, com valores de 2,12 segundos para instâncias menores (como E-n22-k4) a 91,49 segundos para instâncias maiores (como B-n56-k7). Esse comportamento indica que, à medida que a complexidade e o número de veículos aumentam, o tempo de processamento também cresce, devido à dificuldade adicional de encontrar rotas otimizadas para demandas mais complexas.

A incorporação da Busca Tabu ao algoritmo ACO foi fundamental para introduzir uma memória de curto prazo, evitando que o processo de otimização ficasse preso em mínimos locais. Essa memória permitiu ao algoritmo explorar diferentes áreas do espaço de soluções, mesmo quando isso significava aceitar soluções temporariamente menos favoráveis, visando encontrar soluções melhores a longo prazo. Esse mecanismo aumentou a qualidade das soluções para pequenas instâncias, demonstrando que a combinação ACO + Busca Tabu é mais eficaz na exploração de soluções variadas.

No entanto, para problemas maiores e mais complexos, o ACO com Busca Tabu apresentou limitações em termos de tempo de convergência. Por essa razão, foi aplicada uma abordagem adicional utilizando Algoritmos Genéticos (GA) para gerar uma população inicial de soluções para o ACO. Essa combinação (GA + ACO + Busca Tabu) mostrou-se mais eficaz na busca de soluções de menor custo, pois o GA auxilia na geração de soluções iniciais mais diversas e de melhor qualidade, reduzindo o tempo necessário para que o ACO + Busca Tabu converja para soluções otimizadas.

Os resultados dessa abordagem híbrida estão apresentados na Tabela 13. A combinação de ACO + Busca Tabu com GA melhorou significativamente os custos finais, aproximando as soluções do custo mínimo possível e superando as soluções obtidas apenas com ACO + Busca Tabu. A inclusão de GA na fase inicial do ACO provou ser uma estratégia eficaz para otimizar o CVRP em instâncias de grande porte, oferecendo uma abordagem viável para resolver problemas de minimização de custos.

Assim, conclui-se que o ACO com Busca Tabu é adequado para instâncias menores, fornecendo soluções eficientes e custo-benefício em prazos aceitáveis. Para instâncias

maiores, a inclusão de Algoritmos Genéticos para geração inicial das rotas representa uma alternativa promissora para alcançar melhores resultados em escalas de tempo viáveis. A próxima etapa do estudo visa explorar ainda mais essa abordagem combinada para resolver o CVRP em larga escala, conforme abordado por (WANG; LU, 2009).

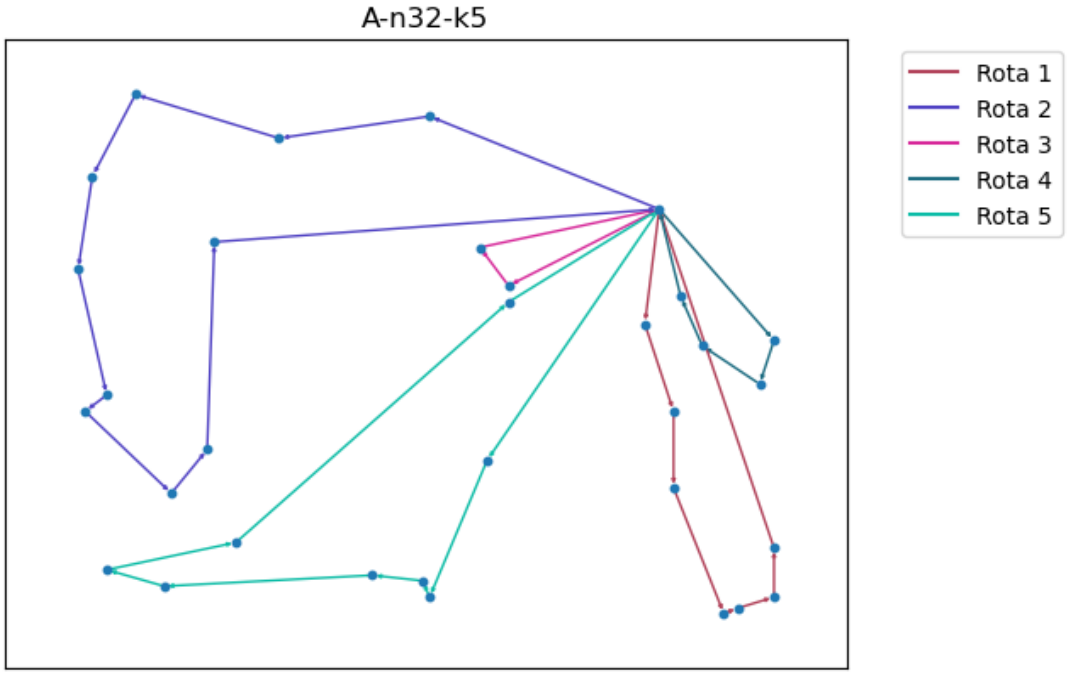

Figura 3 – Gráfico de solução para A-n32-k5

Rota	Cliente
1	0, 26, 7, 13, 17, 19, 31, 21
2	0, 20, 5, 25, 10, 15, 22, 9, 8, 18, 29
3	0, 24, 27
4	0, 12, 1, 16, 30
5	0, 6, 2, 3, 23, 4, 11, 28, 14

Tabela 11 – Rotas dos veículos para A-n32-k5

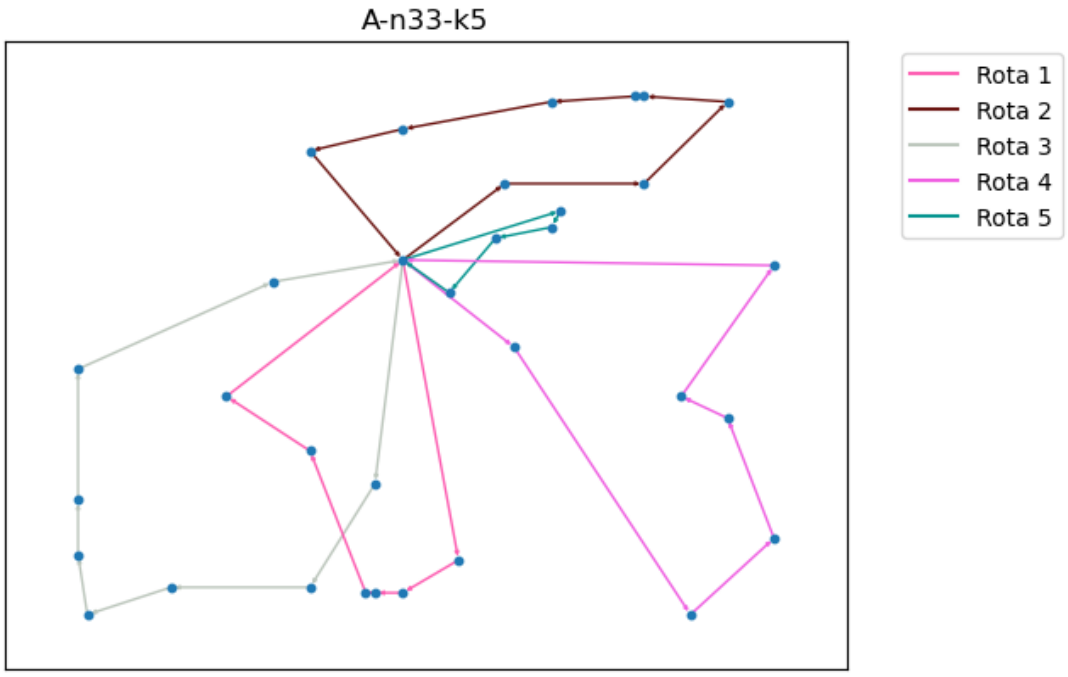

Figura 4 – Gráfico de solução para A-n33-k5

Rota	Cliente
1	0, 10, 30, 25, 27, 4, 20
2	0, 11, 31, 1, 21, 14, 19, 6, 24
3	0, 12, 5, 26, 7, 8, 13, 32, 2
4	0, 15, 17, 9, 3, 16, 29
5	0, 18, 28, 23, 22

Tabela 12 – Rotas dos veículos para A-n33-k5

As Figuras 3 e 4 ilustram visualmente as melhores rotas encontradas pelo algoritmo ACO Tabu para os problemas A-n32-k5 e A-n33-k5, respectivamente. Elas mostram a trajetória do veículo desde o depósito, passando por todos os pontos necessários e retornando ao depósito. Esta representação gráfica demonstra as rotas otimizadas dos algoritmos considerando as restrições do problema. As rotas geradas para cada veículo podem ser revisadas nas Tabelas 11 e 12.

As tabelas 13, 14, e 15 apresentam dados relacionados a diferentes instâncias do Problema de Roteamento de Veículos com Capacidade (CVRP) resolvidos usando um Algoritmo Genético (GA) combinado com a Otimização por Colônia de Formigas (ACO). Cada linha corresponde a uma instância específica do problema identificada pelo seu nome. Os dados incluem o número de veículos utilizados (Num Vehicles), o custo da solução encontrada (Cost), o tempo de execução do algoritmo (Time), o número de iterações realizadas (Iterations), bem como métricas como o valor ótimo encontrado para a instância, a diferença do custo ótimo obtido pelo GA-ACO+Tabu.

Esses dados são cruciais para avaliar o desempenho dos algoritmos GA-ACO+Tabu e ACO na solução do CVRP. Eles fornecem informações sobre a qualidade das soluções

obtidas, o tempo de execução dos algoritmos e a variabilidade dos resultados entre diferentes execuções. Esses insights permitem a comparação e análise do desempenho dos algoritmos em diferentes instâncias do problema, auxiliando na seleção da abordagem mais adequada para cada cenário.

Os campos da tabela são descritos da seguinte forma:

- **Valor Ótimo**: Este campo representa o valor ótimo encontrado para a instância específica do problema. O valor ótimo é o custo mínimo ou máximo, dependendo do objetivo da otimização, alcançado após a execução do algoritmo.
- **Nome da Instância**: Este campo fornece o nome da instância do problema sendo analisada. Geralmente inclui um identificador único ou uma descrição que ajuda a distinguir entre diferentes conjuntos de dados ou problemas.
- **Número de veículos**: Indica o número de veículos utilizados na instância do problema. Esse valor pode variar dependendo da complexidade da instância e das características específicas do problema.
- **Custo**: Representa o custo associado à solução encontrada pelo algoritmo. O custo pode ser uma métrica de desempenho, como o custo operacional total, e é uma das principais medidas da eficácia dos algoritmos de otimização.
- **Tempo**: Este campo mostra o tempo que o algoritmo levou para encontrar a solução para a instância específica. O tempo é tipicamente medido em horas, minutos e segundos, e é crucial para avaliar a eficiência do algoritmo.
- **Iteração**: Indica o número de iterações realizadas pelo algoritmo durante a busca pela solução ótima. Mais iterações podem indicar uma busca mais aprofundada, mas também podem aumentar o tempo de execução.
- **Diferença do custo ótimo**: Este campo apresenta a diferença do custo ótimo das soluções obtidas pelo algoritmo GA-ACO+Tabu para a instância específica.A diferença do custo ótimo é uma medida da variabilidade dos resultados, indicando a dispersão das soluções em torno da média.

		GA-ACO+Tabu					**ACO+Tabu**			
Valor Ótimo	Nome da Instância	Nº de Veículos	Custo	Tempo	Iterações	Diferença do custo ótimo (GA-ACO+Tabu)	Nº de Veículos	Custo	Iterações	Diferença do custo ótimo (ACO+Tabu)
784	A-n32-k5	5	784	00:00:19	3	0.00	5	784.00	4	0.00
661	A-n33-k5	5	661	00:00:22	3	0.00	5	661.00	2	0.00
742	A-n33-k6	6	742	00:00:28	2	0.00	6	742.00	4	0.00
778	A-n34-k5	5	778	00:00:27	3	0.00	5	786.00	2	8.00
799	A-n36-k5	5	799	00:01:00	2	0.00	5	813.00	2	14.00
669	A-n37-k5	5	669	00:01:09	2	0.00	5	669.00	2	0.00
730	A-n38-k5	5	731	00:00:11	3	1.00	5	730.00	2	0.00
822	A-n39-k5	5	822	00:01:12	3	0.00	6	832.00	2	10.00
937	A-n44-k6	6	953	00:00:12	3	16.00	6	958.00	2	21.00
1146	A-n45-k7	7	1164	00:01:59	2	18.00	7	1154.00	4	8.00
1354	A-n60-k9	9	1364	00:00:50	5	10.00	9	1366.00	5	12.00
1034	A-n61-k9	10	1045	00:00:32	4	11.00	10	1049.00	3	15.00
1288	A-n62-k8	8	1319	00:00:56	5	31.00	8	1313.00	3	25.00
1314	A-n63-k10	10	1334	00:00:32	4	20.00	10	1327.00	4	13.00
1616	A-n63-k9	10	1638	00:00:27	3	22.00	10	1651.00	6	35.00
1401	A-n64-k9	9	1438	00:00:37	2	37.00	9	1432.00	4	31.00
1174	A-n65-k9	9	1181	00:00:48	6	7.00	9	1189.00	3	15.00
1159	A-n69-k9	9	1182	00:00:51	4	23.00	9	1176.00	4	17.00
1763	A-n80-k10	10	1808	00:02:36	3	45.00	10	1793.00	9	30.00

Tabela 13 – Resultados das instâncias A do algoritmo ACO+Tabu comparados com GA+(ACO+Tabu)

		GA-ACO+Tabu					**ACO+Tabu**			
Valor Ótimo	Nome	Nº de Veículos	Custo	Tempo	Iterações	Diferença do custo ótimo (GA-ACO+Tabu)	Nº de Veículos	Custo	Iterações	Diferença do custo ótimo (ACO+Tabu)
788	B-n34-k5	5	788	00:00:21	2	0.00	5	788	3	0.00
955	B-n35-k5	5	955	00:01:04	2	0.00	5	955	2	0.00
829	B-n41-k6	6	829	00:02:13	2	0.00	6	834	5	2.50
742	B-n43-k6	6	742	00:03:04	2	0.00	6	745	2	2.12
909	B-n44-k7	7	909	00:05:41	3	0.00	7	912	3	2.12
741	B-n50-k7	7	741	00:00:13	2	0.00	7	741	2	0.00
1312	B-n50-k8	8	1325	00:13:43	3	0.00	8	1327	2	1.41
1032	B-n51-k7	8	1016	00:01:55	2	8.00	8	1016	4	8.00
747	B-n57-k7	8	1140	00:01:32	2	9.13	8	1140	2	9.13
1598	B-n57-k9	9	1608	00:02:12	2	7.07	9	1616	3	12.73
1316	B-n66-k9	9	1330	00:00:35	3	7.07	9	1323	2	4.95
1272	B-n68-k9	9	1294	00:02:05	6	11.00	9	1299	2	14.14
1221	B-n78-k10	10	1247	00:08:05	4	13.86	10	1256	2	17.04

Tabela 14 – Resultados das instâncias B dos algoritmos ACO+Tabu comparados com GA+(ACO+Tabu)

Os resultados das instâncias para os algoritmos ACO+Tabu e AG+(ACO+Tabu) são apresentados nas Tabelas 13 e 14 para diferentes conjuntos de dados (instâncias A e B). Pode-se observar que o número de veículos utilizados é consistente entre os dois algoritmos, variando de 5 a 10, e os custos das soluções obtidas são competitivos em relação aos valores ótimos conhecidos. Para a instância A, o custo variou de 669 a 1808 para AG-ACO+Tabu e de 669 a 1793 para ACO. O tempo de execução para ACO foi geralmente mais curto, variando de 4 segundos a 1 minuto e 44 segundos, enquanto AG-ACO+Tabu teve tempos variando de 8 segundos a 2 minutos e 36 segundos.

Para a instância B, o número de veículos variou de 5 a 10, com custos variando de 741 a 1608 para AG-ACO+Tabu e de 741 a 1616 para ACO. Novamente, o tempo de execução do ACO foi mais curto, variando de 5 a 32 segundos, enquanto AG-ACO+Tabu variou de 10 segundos a 8 minutos e 5 segundos. A diferença do custo ótimo para ambas as instâncias foi próximo de zero, indicando consistência na obtenção de resultados ótimos pelos algoritmos.

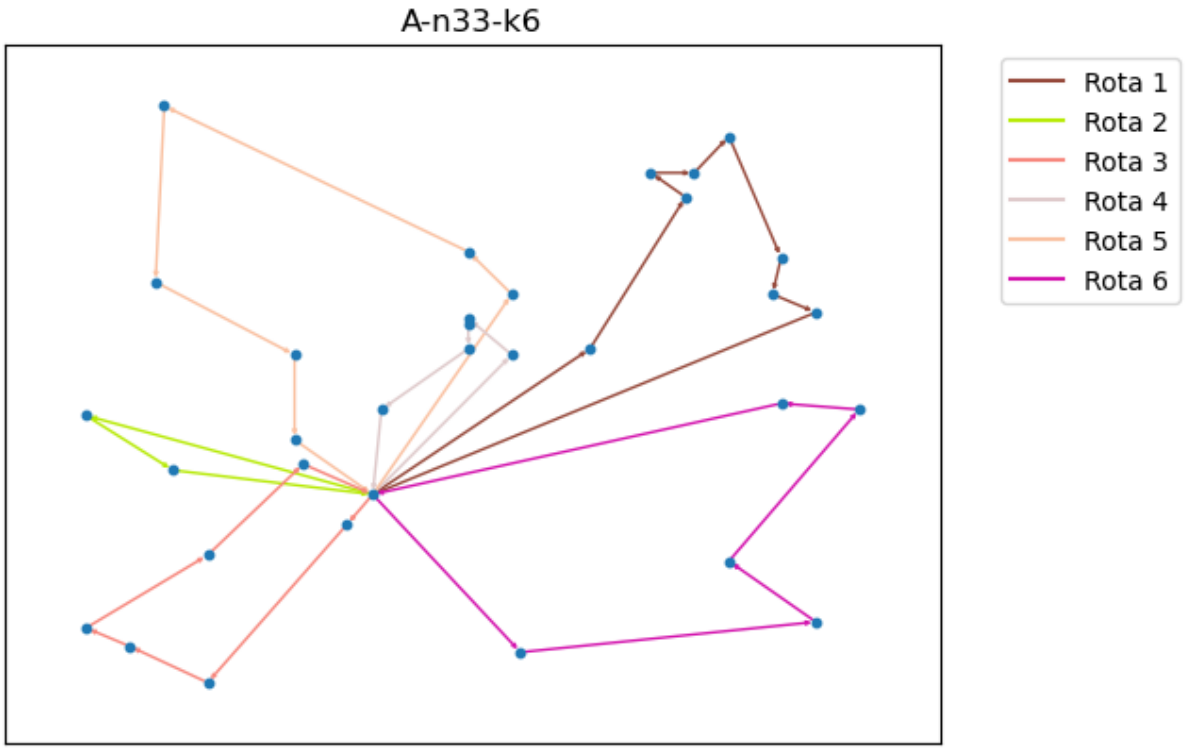

Figura 5 – Gráfico da solução para A-n33-k6

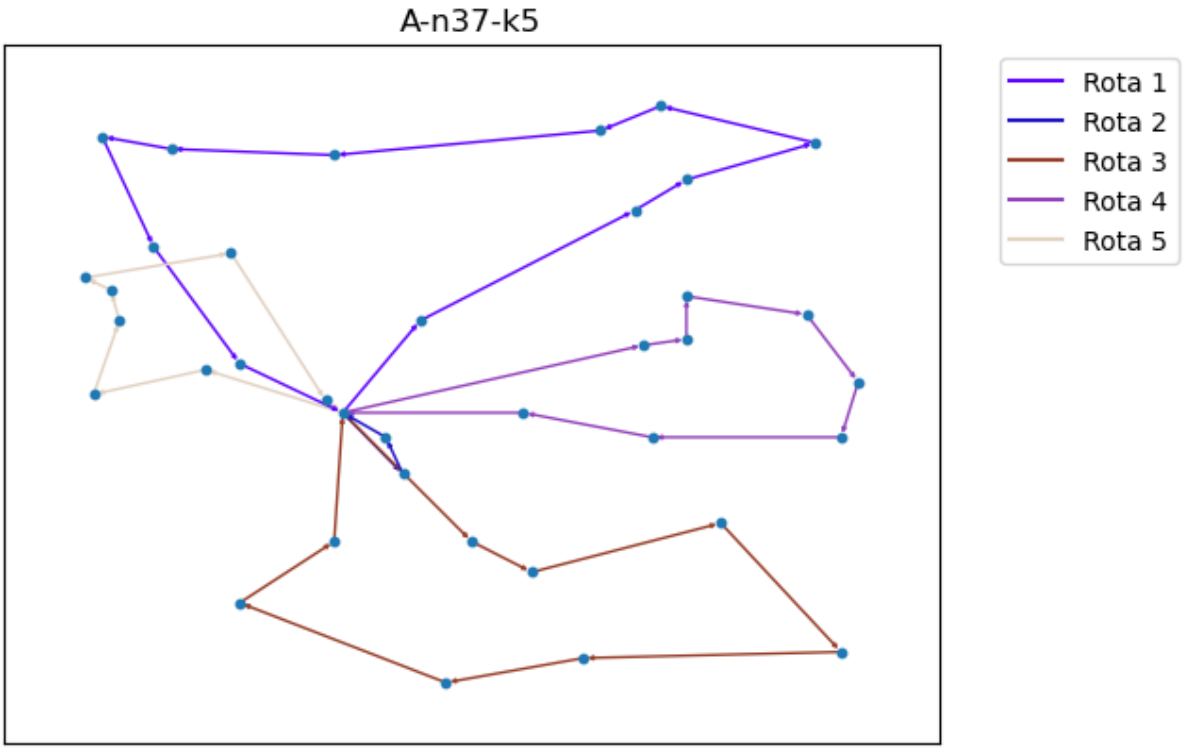

Figura 6 – Gráfico da solução para A-n37-k5

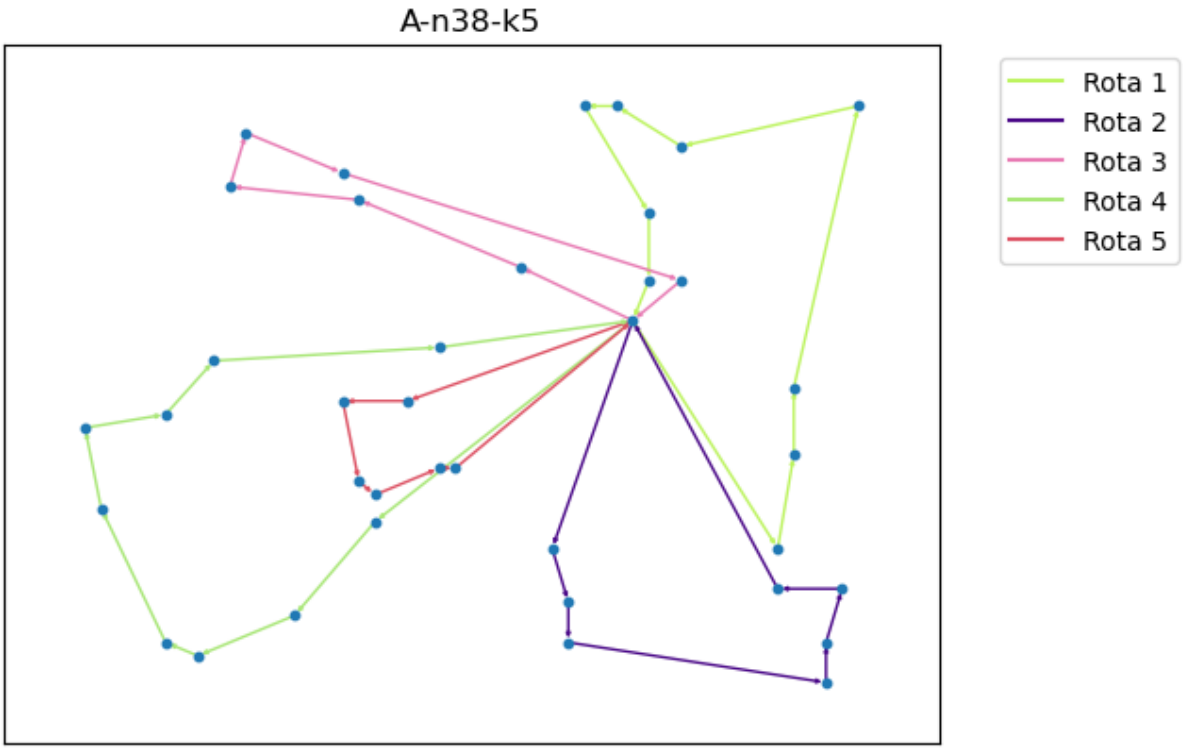

Figura 7 – Gráfico da solução para A-n38-k5

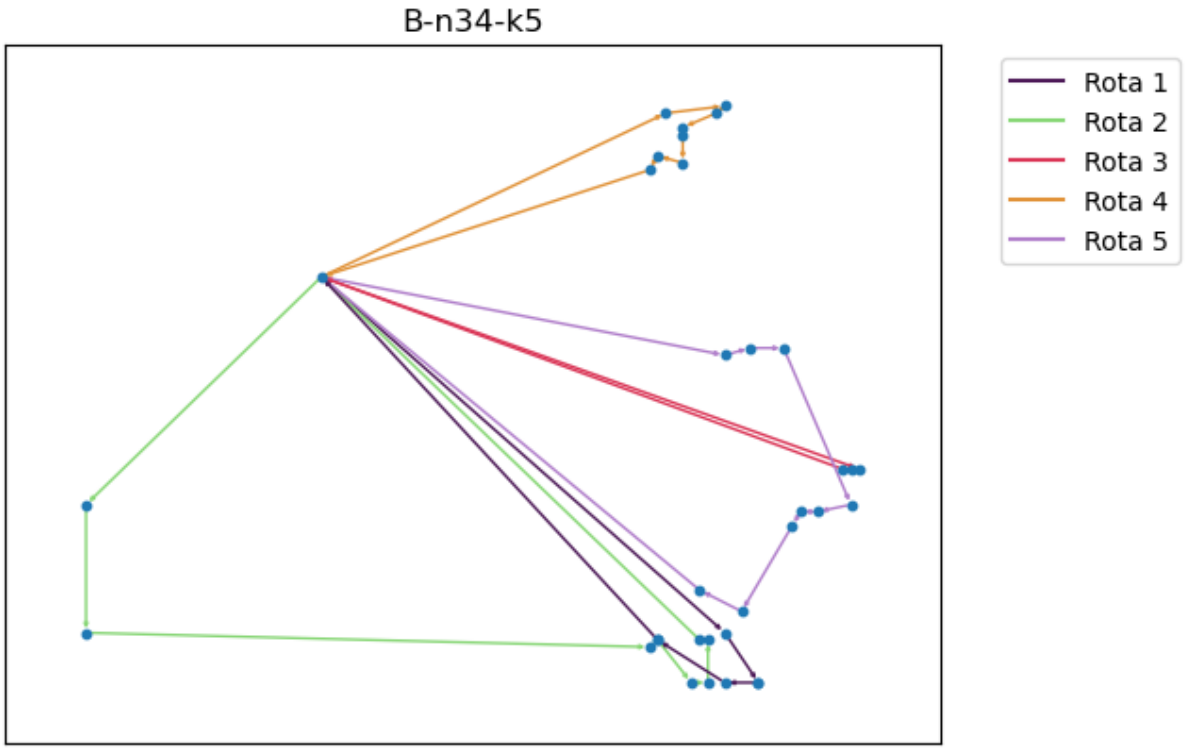

Figura 8 – Gráfico da solução para B-n34-k5

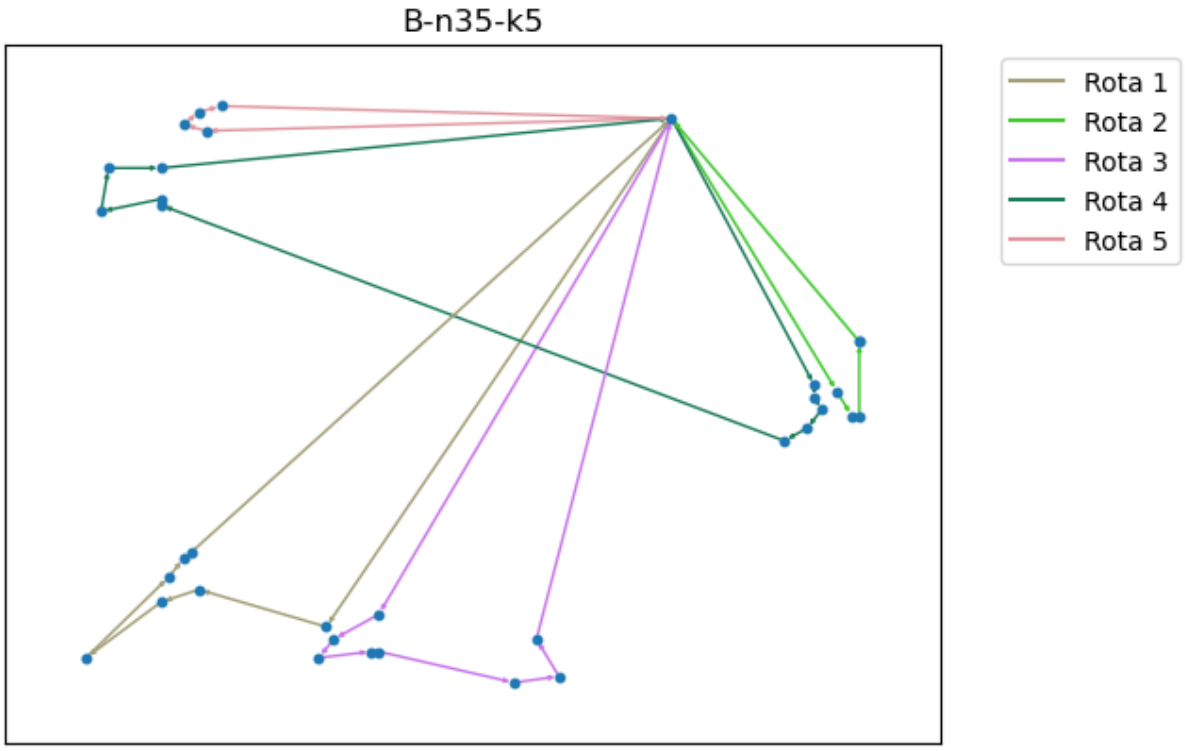

Figura 9 – Gráfico da solução para B-n35-k5

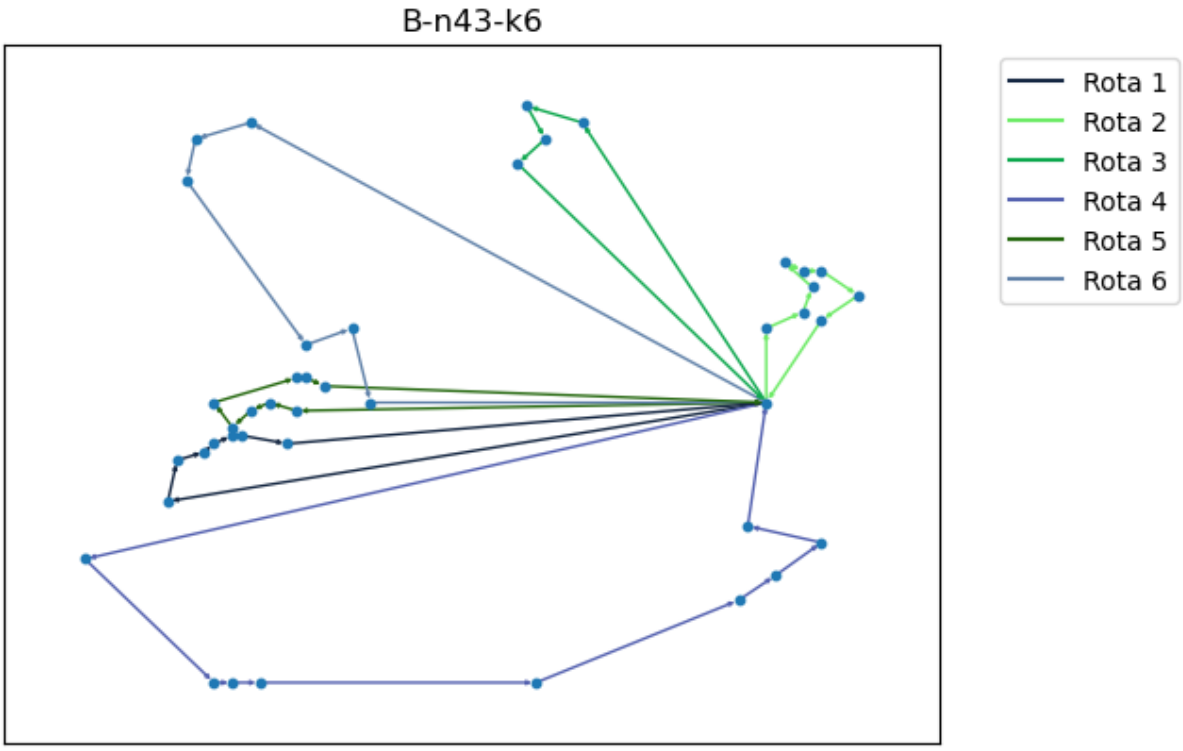

Figura 10 – Gráfico da solução para B-n43-k6

As Figuras 5, 6, 7, 8, 9, e 10 mostram a distribuição das rotas pelos veículos para algumas instâncias A e B, que foram selecionadas para representar os melhores resultados. Quando os resultados foram comparados com os valores ótimos conhecidos, os custos das soluções obtidas por ACO+Tabu e AG+(ACO+Tabu) foram muito próximos ou iguais aos valores ótimos, destacando a eficácia dos algoritmos em encontrar soluções de alta qualidade. O ACO mostrou-se mais rápido e consistente, enquanto o AG+(ACO+Tabu) alcançou soluções ligeiramente mais eficientes em termos de custo em algumas instâncias. Portanto, a escolha entre os dois algoritmos pode depender de necessidades específicas do problema, como velocidade de execução ou a busca pela solução de menor custo possível.

Para as instâncias de Li, onde foram obtidos os melhores resultados, a Tabela 15 apresenta os resultados de instâncias de alta complexidade resolvidas pelos algoritmos AG+ACO+Tabu. Essas instâncias do CVRP são de tamanhos significativos e não possuem valores ótimos estabelecidos na literatura. No entanto, os valores ótimos encontrados por (LI; GOLDEN; WASIL, 2005) são usados como base para comparação.

Melhor Valor (LI; GOLDEN; WASIL, 2005)	Nome	Número de Veículos	Melhor Encontrado	Tempo Encontrado	Diferença do custo ótimo
16212.83	Li_21	10	16689.0	05:55:59	1476.17
14499.04	**Li_22**	**14**	**14133.0**	**09:11:24**	**29.010**
18801.13	Li_23	10	19312.0	08:05:56	81.05
21389.43	Li_24	10	22153.0	24:03:00	763.57
16665.7	**Li_25**	**10**	**15183.0**	**24:19:00**	**1482.70**
23977.73	**Li_26**	**10**	**23861**	**28:35:00**	**116.73**

Tabela 15 – Resultados para instâncias L usando o algoritmo ACO+Tabu comparado com AG+(ACO+Tabu)

As instâncias demostradas nas Figuras 11, 12 e 13, respectivamente, que superaram os valores ótimos de referência, mostram a distribuição eficiente dos veículos em cada rota. apresentaram bons resultados com o algoritmo AG+ACO+Tabu, superando os valores ótimos de referência. Para a instância Li_22, o valor ótimo de referência é 14499.04, e a melhor solução encontrada foi 14133.0, alcançada em 9 horas, 11 minutos e 24 segundos usando 14 veículos. Esta solução é melhor que o valor ótimo de referência. Para a instância Li_25, o valor ótimo de referência é 16665.7, e a melhor solução encontrada foi 15183.0. Esta solução é significativamente melhor que o valor ótimo, indicando a alta eficiência do algoritmo. Na instância Li_26, o valor ótimo de referência é 23977.73, e a melhor solução encontrada foi 23861, com um tempo de execução de 28 horas e 35 minutos usando 10 veículos. Esta solução está abaixo do valor ótimo, mostrando um excelente desempenho, apesar do tempo de execução extremamente alto, refletindo a alta complexidade do problema.

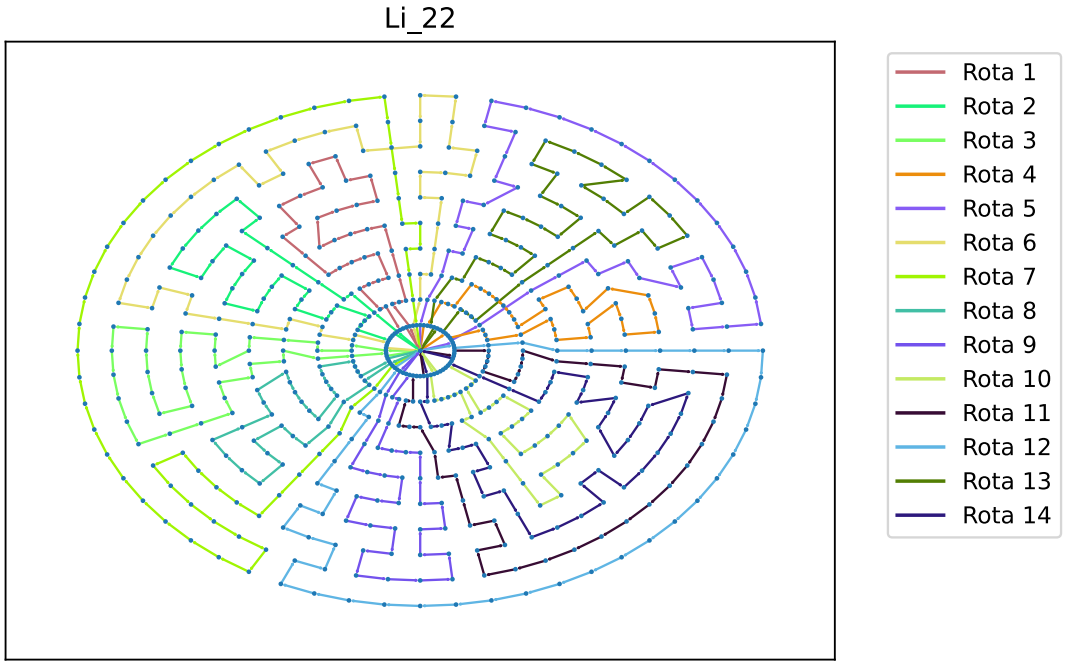

Figura 11 – Gráfico da solução para Li-22

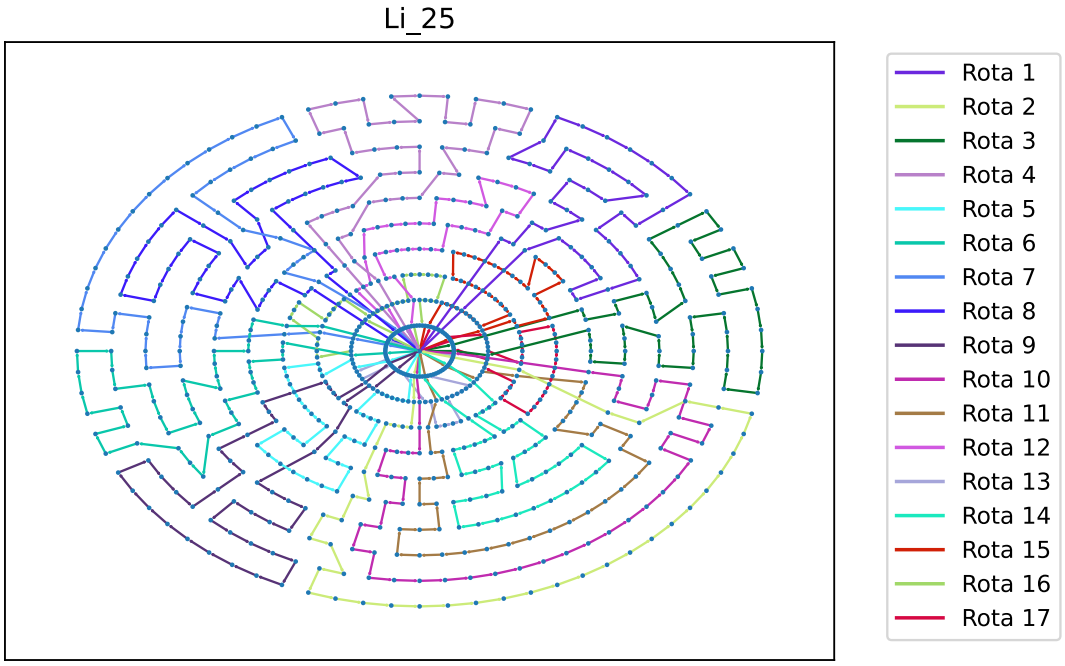

Figura 12 – Gráfico da solução para Li-25

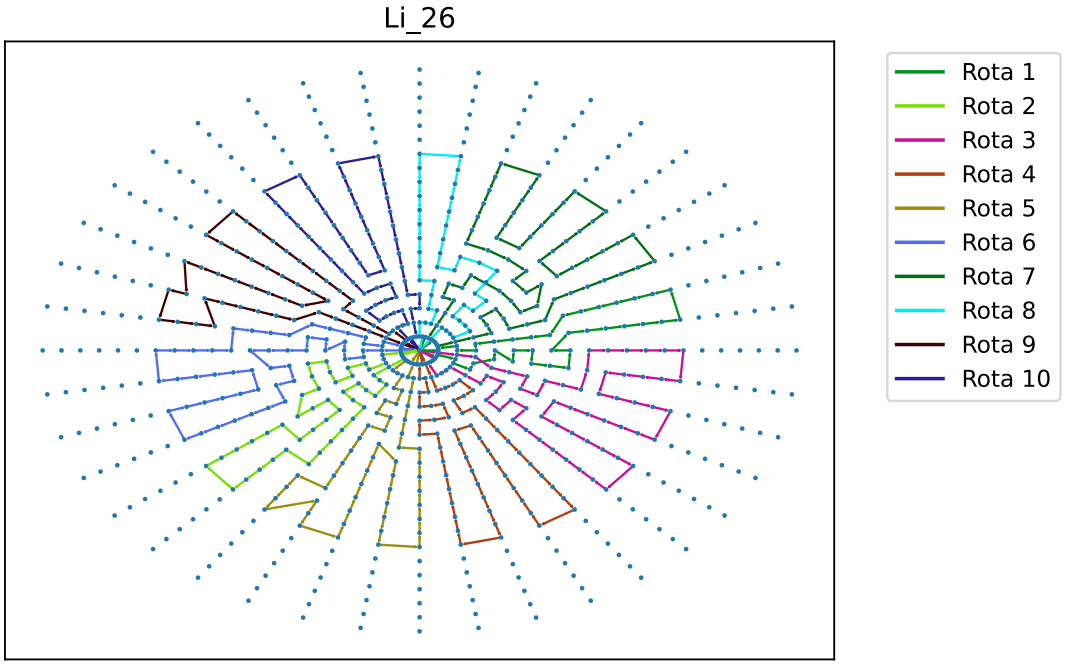

Figura 13 – Gráfico da solução para Li-26

Para a instância Li_22, os veículos foram alocados de maneira a cobrir as áreas necessárias de forma eficaz, resultando em uma solução melhor que o valor ótimo. Para a instância Li_25, a visualização das rotas revela uma organização otimizada que alcançou uma solução significativamente melhor que o valor ótimo, demonstrando a eficiência do algoritmo na distribuição das rotas. Da mesma forma, para a instância Li_26, apesar dos longos tempos de execução, a solução conseguiu superar o valor ótimo de referência. Essas figuras demonstram a capacidade do algoritmo de otimizar

a alocação de veículos, resultando em soluções eficazes para problemas complexos de CVRP.

Capítulo 5

CONCLUSÃO

Este estudo apresenta uma abordagem híbrida de metaheurísticas que combina Algoritmos Genéticos (GA), Otimização por Colônia de Formigas (ACO) e Busca Tabu (TS) para resolver eficazmente instâncias complexas do Problema de Roteamento de Veículos com Capacidade (CVRP). O objetivo é desenvolver uma solução que minimize os custos operacionais, ao mesmo tempo em que assegura a otimização eficiente das rotas no roteamento de veículos.

Os resultados demonstram que o algoritmo híbrido GA+ACO+TS produz soluções próximas do ótimo ou, em alguns casos, supera valores próximos ao ótimo. Essa performance evidencia a eficiência do método em lidar com as complexas demandas do CVRP, destacando seu potencial como uma ferramenta poderosa para enfrentar os desafios logísticos associados ao roteamento de veículos.

Embora os tempos de execução variem entre as diferentes instâncias devido à complexidade inerente do problema, a baixa diferença do custo ótimo na qualidade das soluções sugere consistência e robustez. Essa adaptabilidade é essencial para aplicações práticas, onde restrições variadas e dinâmicas frequentemente surgem.

A sinergia entre GA, ACO e TS desempenha um papel fundamental no sucesso do algoritmo. O GA contribui com capacidades de busca evolutiva, o ACO se destaca na exploração de espaços de solução inspirados por processos naturais, e o TS evita eficientemente ótimos locais através de uma exploração estratégica. Em conjunto, esses elementos permitem que o algoritmo navegue por grandes e complexos espaços de busca de forma eficiente.

Em síntese, o algoritmo híbrido GA+ACO+TS oferece uma base sólida para futuras pesquisas em otimização de roteamento de veículos. Ele não apenas proporciona soluções de alta qualidade, mas também atende às demandas computacionais de instâncias de CVRP em larga escala.

5.1 Dificuldades encontradas

No desenvolvimento[1] e aplicação do algoritmo híbrido GA+ACO+TS para o Problema de Roteamento de Veículos com Capacidade (CVRP), diversas dificuldades significativas foram identificadas. A principal delas foi a complexidade computacional associada à resolução de instâncias de grande escala. Embora o algoritmo tenha demonstrado bom desempenho na qualidade das soluções, o aumento no tamanho das instâncias, com mais veículos e pontos de entrega, resultou em tempos de execução elevados. Essa complexidade exigiu ajustes finos nos parâmetros do algoritmo para equilibrar a qualidade das soluções com a eficiência computacional.

Outra dificuldade relevante foi a definição dos parâmetros ideais para os três algoritmos envolvidos — Algoritmo Genético (GA), Otimização por Colônia de Formigas (ACO) e Busca Tabu (TS). A escolha desses parâmetros impacta diretamente o desempenho do modelo, sendo necessário realizar uma série de testes para encontrar as configurações mais adequadas, equilibrando a exploração do espaço de soluções e a obtenção de soluções de alta qualidade. A diversidade das instâncias do problema também apresentou um desafio, pois características variadas, como o número de veículos e restrições de capacidade, exigiram ajustes específicos no algoritmo para garantir a adaptação eficiente.

5.2 Trabalhos Futuros

A análise e o desenvolvimento do algoritmo híbrido GA+ACO+TS para o CVRP abrem várias possibilidades de investigação futura. Uma das principais direções a serem exploradas é o aprimoramento da eficiência computacional. Embora o algoritmo tenha se mostrado eficaz na qualidade das soluções, o tempo de execução para instâncias de grande escala ainda representa um desafio. Futuras pesquisas podem explorar técnicas como paralelização, uso de unidades de processamento gráfico (GPUs) ou o desenvolvimento de algoritmos híbridos mais sofisticados, com o intuito de reduzir a complexidade computacional sem comprometer a qualidade das soluções.

Outra linha promissora seria a implementação de ajuste automático de parâmetros. A definição dos parâmetros dos algoritmos de busca evolutiva, colônia de formigas e busca tabu é crucial para o desempenho do modelo. Nesse sentido, o uso de métodos de otimização de parâmetros, como algoritmos de busca hiperparamétrica ou aprendizado de máquina, pode melhorar a performance automatizando o processo de definição dos melhores valores para cada parâmetro.

A integração de outras metaheurísticas também apresenta uma possibilidade interessante. A combinação do algoritmo híbrido com técnicas como Simulated Annealing (SA) ou Algoritmos de Colônia de Abelha pode expandir a capacidade do modelo de explorar diferentes espaços de solução e melhorar seu desempenho em instâncias de

[1] <https://github.com/gferreiracunha123/CRVP-ACO>

maior complexidade. Testes em cenários dinâmicos, que envolvem mudanças nas condições do problema durante a execução, como a inserção de novas demandas ou a alteração das restrições, também representam uma linha de pesquisa promissora. Tais abordagens poderiam aumentar a aplicabilidade do algoritmo em problemas reais, onde as condições podem variar ao longo do tempo.

Por fim, uma investigação em aplicações práticas nas áreas de logística e transporte é essencial. A validação do algoritmo em cenários do mundo real, como o roteamento de frotas de veículos para entregas urbanas ou a gestão de veículos de serviços públicos, pode fornecer insights valiosos sobre a viabilidade do método e possíveis adaptações necessárias para aprimorar sua aplicação em contextos logísticos.

Referências

BELLO, I. et al. **Neural Combinatorial Optimization with Reinforcement Learning**. 2017.

BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 35, n. 3, p. 268–308, sep 2003. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/937503.937505>.

CRUZ, H. F. A. da; CUNHA, A. Salles da. The profitable single truck and trailer routing problem with time windows: Formulation, valid inequalities and branch-and-cut algorithms. **Computers & industrial engineering**, Elsevier Ltd, v. 180, p. 109238, 2023. ISSN 0360-8352.

DENG, W. et al. A novel two-stage hybrid swarm intelligence optimization algorithm and application. **Soft Computing**, Springer Science and Business Media LLC, v. 16, n. 10, p. 1707–1722, maio 2012. ISSN 1433-7479. Disponível em: <https://doi.org/10.1007/s00500-012-0855-z>.

DORIGO, M.; L.M., G. Ant colonies for the travelling salesman problem. **Biosystems**, 1997.

ELNEIMA, A.; SALIH, M. Optimisation of vehicle routing problem using hyper-heuristics. In: **2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)**. [S.l.: s.n.], 2021. p. 1–7.

FAIZ, S.; KRICHEN, S.; INOUBLI, W. A dss based on gis and tabu search for solving the cvrp: The tunisian case. **The Egyptian Journal of Remote Sensing and Space Science**, v. 17, n. 1, p. 105–110, 2014. ISSN 1110-9823. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1110982313000288>.

GALINDO, J. C. F. et al. A multi-agent system for solving the dynamic capacitated vehicle routing problem with stochastic customers using trajectory data mining. **CoRR**, arXiv, abs/2009.12691, 2020. Disponível em: <https://arxiv.org/abs/2009.12691>.

GLOVER F.;KOCHENBERGER, G. **Handbook of metaheuristics**. [S.l.]: Boston:Kluwer Academic Publishers, 2003.

GOLDBARG; LUNA. **Otimização Combinatória e MetaHeurísticas: Algoritmos e Aplicações**. [S.l.]: Elsevier, 2016.

GUEDES, A. d. C. B.; LEITE, J. N. F.; ALOISE, D. J. Um algoritmo genético com infecçao viral para o problema do caixeiro viajante. **Revista PublICa**, v. 1, n. 1, out. 2009. Disponível em: <https://periodicos.ufrn.br/publica/article/view/125>.

HADDADENE, S. R. A.; LABADIE, N.; PRODHON, C. A GRASP× ILS for the vehicle routing problem with time windows, synchronization and precedence constraints. **Expert Systems with Applications**, Elsevier, v. 66, p. 274–294, 2016.

HENDRAWAN, R.; BAIZAL, Z.; WULANDARI, G. S. Generating a multi-day travel itinerary recommendation using the hybrid ant colony system and brainstorm optimization algorithm. **International Journal of Intelligent Engineering & Systems**, v. 17, n. 2, 2024.

HOLLAND, J. H. **Adaptation in natural and artificial systems**. [S.l.]: University of Michigan Press, 1975.

JIA, Y.-H.; MEI, Y.; ZHANG, M. A bilevel ant colony optimization algorithm for capacitated electric vehicle routing problem. **IEEE Transactions on Cybernetics**, Institute of Electrical and Electronics Engineers (IEEE), p. 1–14, 2021. Disponível em: <https://doi.org/10.1109/tcyb.2021.3069942>.

KIRKPATRICK, S.; JR, C. D. G.; VECCHI, M. P. Optimization by simulated annealing. **science**, American association for the advancement of science, v. 220, n. 4598, p. 671–680, 1983.

KOOL, W.; HOOF, H. van; WELLING, M. **Attention, Learn to Solve Routing Problems!** 2019.

LAPORTE, G. The vehicle routing problem: An overview of exact and approximate algorithms. **European Journal of Operational Research**, Elsevier BV, v. 59, n. 3, p. 345–358, 1992. ISSN 0377-2217. Disponível em: <https://www.sciencedirect.com/science/article/pii/037722179290192C>.

LI, F.; GOLDEN, B.; WASIL, E. Very large-scale vehicle routing: new test problems, algorithms, and results. **Computers & Operations Research**, v. 32, n. 5, p. 1165–1179, 2005. ISSN 0305-0548. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0305054803003150>.

LI, J. et al. Deep reinforcement learning for solving the heterogeneous capacitated vehicle routing problem. **IEEE Transactions on Cybernetics**, Institute of Electrical and Electronics Engineers (IEEE), PP, n. 12, p. 13572–13585, 09 2021. ISSN 2168-2275.

____. Deep reinforcement learning for solving the heterogeneous capacitated vehicle routing problem. **IEEE transactions on cybernetics**, IEEE, Ithaca, v. 52, n. 12, p. 1–14, 2022. ISSN 2168-2267.

LINDHOLM, M.; WIEDING, S. von. Challenges in urban freight transport planning - a review in the baltic sea region. **Journal of Transport Geography**, Elsevier BV, v. 22, p. 129–136, 05 2012. ISSN 0966-6923.

MACHADO, A. M. et al. A new hybrid matheuristic of GRASP and VNS based on constructive heuristics, set-covering and set-partitioning formulations applied to the capacitated vehicle routing problem. **Expert systems with applications**, Elsevier Ltd, New York, v. 184, p. 115556, 2021. ISSN 0957-4174.

MARTÍ, R.; REINELT, G. Heuristic methods. In: **Exact and Heuristic Methods in Combinatorial Optimization: A Study on the Linear Ordering and the Maximum Diversity Problem**. [S.l.]: Springer, 2022. p. 27–57.

MINGPRASERT, S.; MASUCHUN, R. Adaptive artificial bee colony algorithm for solving the capacitated vehicle routing problem. In: **International Conference on Knowledge and Smart Technology (KST)**. [S.l.: s.n.], 2017. p. 23–27.

NAZARI, M. et al. **Reinforcement Learning for Solving the Vehicle Routing Problem**. 2018.

OBAID, O. I. Solving capacitated vehicle routing problem (CVRP) using Tabu Search Algorithm (TSA). **Ibn AL-Haitham Journal For Pure and Applied Sciences**, University of Baghdad, v. 31, n. 2, p. 199–209, 2018. ISSN 1609-4042.

PAPADIMITRIOU, C. H.; STEIGLITZ, K. **Combinatorial Optimization : Algorithms and Complexity**. Dover Publications, 1998. Paperback. ISBN 0486402584. Disponível em: <http://www.amazon.fr/exec/obidos/ASIN/0486402584/citeulike04-21>.

SOUZA, C. d. O. et al. Soluções para o transporte urbano de cargas na etapa de última milha. **urbe. Revista Brasileira de Gestão Urbana**, Pontifícia Universidade Católica do Paraná, v. 12, p. e20190138, 2020. ISSN 2175-3369. Disponível em: <https://doi.org/10.1590/2175-3369.012.e20190138>.

TAILLARD Éric D.; HELSGAUN, K. Popmusic for the travelling salesman problem. **European Journal of Operational Research**, v. 272, n. 2, p. 420–429, 2019. ISSN 0377-2217. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0377221718305745>.

TOTH, P. et al. **Vehicle Routing: Problems, Methods, and Applications**. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2015. (MOS-SIAM series on optimization). ISBN 9781523109371. Disponível em: <https://books.google.com.br/books?id=YL_CswEACAAJ>.

VINYALS, O.; FORTUNATO, M.; JAITLY, N. Pointer networks. In: CORTES, C. et al. (Ed.). **Advances in Neural Information Processing Systems**. Curran Associates, Inc., 2015. v. 28. Disponível em: <https://proceedings.neurips.cc/paper/2015/file/29921001f2f04bd3baee84a12e98098f-Paper.pdf>.

WANG, C.-H.; LU, J.-Z. A hybrid genetic algorithm that optimizes capacitated vehicle routing problems. **Expert systems with applications**, Elsevier Ltd, OXFORD, v. 36, n. 2, p. 2921–2936, 2009. ISSN 0957-4174.

WOLFINGER, D.; SALAZAR-GONZáLEZ, J.-J. The pickup and delivery problem with split loads and transshipments: A branch-and-cut solution approach. **European journal of operational research**, Elsevier B.V, v. 289, n. 2, p. 470–484, 2021. ISSN 0377-2217.

ZULFIQAR, L. O. M.; ISNANTO, R. R.; NURHAYATI, O. D. Using CVRP model in designing decision support system for optimizing distribution route and amounts of utilized vehicles. In: **2018 International Conference on Information and Communications Technology (ICOIACT)**. [S.l.: s.n.], 2018. p. 789–792.