



UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS TRINDADE
PROGRAMA DE GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO

Leonardo Seishi Yamazaki

**PROBLEMA DO CAIXEIRO VIAJANTE PARA VEÍCULOS ELÉTRICOS COM
REABASTECIMENTO**

Florianópolis, Santa Catarina – Brasil
2025

Leonardo Seishi Yamazaki

**PROBLEMA DO CAIXEIRO VIAJANTE PARA VEÍCULOS ELÉTRICOS COM
REABASTECIMENTO**

Trabalho de Conclusão de Curso submetido ao Programa de Graduação em Ciências da Computação da Universidade Federal de Santa Catarina para a obtenção do Grau de Bacharel em Ciências da Computação.

Orientador(a): Prof. Rafael de Santiago, Dr.

Florianópolis, Santa Catarina – Brasil

2025

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.
Dados inseridos pelo próprio autor.

Yamazaki, Leonardo Seishi

PROBLEMA DO CAIXEIRO VIAJANTE PARA VEÍCULOS ELÉTRICOS
COM REABASTECIMENTO / Leonardo Seishi Yamazaki ;
orientador, Rafael de Santiago, 2026.
59 p.

Trabalho de Conclusão de Curso (graduação) -
Universidade Federal de Santa Catarina, Centro Tecnológico,
Graduação em Ciências da Computação, Florianópolis, 2026.

Inclui referências.

1. Ciências da Computação. 2. Metaheurística. 3.
Algoritmos. 4. Problema do Caixeiro Viajante. 5. Veículos
elétricos. I. Santiago, Rafael de. II. Universidade
Federal de Santa Catarina. Graduação em Ciências da
Computação. III. Título.

Leonardo Seishi Yamazaki

**PROBLEMA DO CAIXEIRO VIAJANTE PARA VEÍCULOS ELÉTRICOS COM
REABASTECIMENTO**

Este(a) Trabalho de Conclusão de Curso foi julgado adequado(a) para obtenção do Título de Bacharel em Ciências da Computação, e foi aprovado em sua forma final pelo Programa de Graduação em Ciências da Computação do INE – Departamento de Informática e Estatística, CTC – Centro Tecnológico da Universidade Federal de Santa Catarina.

Florianópolis, Santa Catarina – Brasil, 05 de dezembro de 2025.

Lúcia Helena Martins Pacheco, Dr.

Coordenador(a) do Programa de Graduação em
Ciências da Computação

Banca Examinadora:

Prof. Rafael de Santiago, Dr.

Orientador(a)
Universidade Federal de Santa Catarina – UFSC

Prof. Thaís Bardini Idalino, Dr.

Universidade Federal de Santa Catarina – UFSC

Prof. Pedro Belin Castellucci, Dr.

Universidade Federal de Santa Catarina – UFSC

RESUMO

Para minimizar as emissões de gases de efeito estufa e reduzir custos operacionais no longo prazo, o setor logístico está adotando veículos elétricos (VEs) de forma crescente. Embora os VEs já ofereçam vantagens econômicas significativas – com redução de até 60% nos custos de energia e manutenção por quilômetro rodado em comparação com veículos convencionais –, os avanços futuros em tecnologia de baterias, infraestrutura de recarga e integração com redes inteligentes trarão novos desafios computacionais. Este trabalho propõe e implementa uma metaheurística híbrida para o Problema do Caixeiro Viajante com Veículos Elétricos (PCV-E). Nossa abordagem irá seguir três aspectos principais: (1) revisão sistemática da literatura sobre soluções para PCV-E, (2) desenvolvimento de uma heurística de roteamento otimizado que considera restrições de recarga, e (3) análise comparativa de desempenho com métodos existentes. Para isso será desenvolvido um algoritmo combinando duas metaheurísticas, Busca em Vizinhança Variável (VNS) e Busca Tabu (TS), e por meio de benchmarks, demonstrar os aspectos positivos em adotar métodos híbridos e uso de diferentes técnicas de resolução para o PCV-E.

Palavras-chaves: Algoritmos. Heurísticas. PCV-E. Otimização.

ABSTRACT

To minimize greenhouse gas emissions and reduce long-term operational costs, the logistics sector is increasingly adopting electric vehicles (EVs). While EVs already offer significant economic advantages - with up to 60% reduction in energy and maintenance costs per kilometer compared to conventional vehicles - future advancements in battery technology, charging infrastructure, and smart grid integration will introduce new computational challenges. This work proposes and implements a hybrid heuristic for the Electric Vehicle Traveling Salesman Problem (E-TSP). Our approach combines Variable Neighborhood Search (VNS) and Tabu Search (TS) metaheuristics to address three key aspects: (1) systematic literature review of E-TSP solutions, (2) development of an optimized routing heuristic considering charging constraints, and (3) comparative performance analysis against existing methods.

Keywords: Algorithms. Heuristics. E-TCP. Optimization.

LISTA DE FIGURAS

Figura 1	–	Vendas globais de veículos elétricos de 2014 a 2024 (em milhões de unidades). Cada cor representa um mercado regional distinto.	9
Figura 2	–	Representação gráfica de uma instância do ETSP.	14

LISTA DE TABELAS

Tabela 1	–	Comparação: Heurísticas Construtivas vs Metaheurísticas	28
Tabela 2	–	Comparação entre os artigos relacionados	35
Tabela 4	–	3P-Heu para instâncias com 20 clientes e 4 estações Roberti e Wen (2016)	42
Tabela 5	–	3P-Heu para instâncias com 20 clientes e 9 estações Roberti e Wen (2016)	42
Tabela 6	–	Resultados de distâncias para instâncias de 20 clientes e 9 estações com solução inicial <i>NN</i>	42
Tabela 7	–	Resultados de distâncias para instâncias de 20 clientes e 4 estações com solução inicial <i>NN</i>	43
Tabela 8	–	Resultados de tempos para instâncias de 20 clientes e 9 estações com solução inicial <i>NN</i>	43
Tabela 9	–	Resultados de tempos para instâncias de 20 clientes e 4 estações com solução inicial <i>NN</i>	43
Tabela 10	–	Resultados de distâncias para instâncias de 20 clientes e 9 estações com solução inicial Random	44
Tabela 11	–	Resultados de distâncias para instâncias de 20 clientes e 4 estações com solução inicial Random	44
Tabela 12	–	Resultados de tempos para instâncias de 20 clientes e 9 estações com solução inicial Random	45
Tabela 13	–	Resultados de tempos para instâncias de 20 clientes e 4 estações com solução inicial Random	45
Tabela 14	–	3P-Heu para instâncias com 150 clientes e 4 estações Roberti e Wen (2016)	46
Tabela 15	–	3P-Heu para instâncias com 150 clientes e 9 estações Roberti e Wen (2016)	46
Tabela 16	–	Resultados de distâncias para instâncias de 150 clientes e 9 estações com solução inicial <i>NN</i>	47
Tabela 17	–	Resultados de distâncias para instâncias de 150 clientes e 4 estações com solução inicial <i>NN</i>	47
Tabela 18	–	Resultados de tempos para instâncias de 150 clientes e 4 estações com solução inicial <i>NN</i>	47
Tabela 19	–	Resultados de tempos para instâncias de 150 clientes e 9 estações com solução inicial <i>NN</i>	47
Tabela 20	–	Resultados de distâncias para instâncias de 150 clientes e 9 estações com solução inicial Random	48
Tabela 21	–	Resultados de distâncias para instâncias de 150 clientes e 4 estações com solução inicial Random	48
Tabela 22	–	Resultados de tempos para instâncias de 150 clientes e 9 estações com solução inicial Random	48
Tabela 23	–	Resultados de tempos para instâncias de 150 clientes e 4 estações com solução inicial Random	48

Tabela 24	– 3P-Heu para instâncias com 200 clientes e 4 estações Roberti e Wen (2016)	49
Tabela 25	– 3P-Heu para instâncias com 200 clientes e 9 estações Roberti e Wen (2016)	49
Tabela 26	– Resultados de distâncias para instâncias de 200 clientes e 9 estações com solução inicial <i>NN</i>	50
Tabela 27	– Resultados de distâncias para instâncias de 200 clientes e 4 estações com solução inicial <i>NN</i>	50
Tabela 28	– Resultados de tempos para instâncias de 200 clientes e 9 estações com solução inicial <i>NN</i>	50
Tabela 29	– Resultados de tempos para instâncias de 200 clientes e 4 estações com solução inicial <i>NN</i>	50
Tabela 30	– Resultados de distâncias para instâncias de 200 clientes e 9 estações com solução inicial Random	51
Tabela 31	– Resultados de distâncias para instâncias de 200 clientes e 4 estações com solução inicial Random	51
Tabela 32	– Resultados de tempos para instâncias de 200 clientes e 9 estações com solução inicial Random	51
Tabela 33	– Resultados de tempos para instâncias de 200 clientes e 4 estações com solução inicial Random	51

SUMÁRIO

1	INTRODUÇÃO	9
1.1	OBJETIVOS	11
1.1.1	OBJETIVO GERAL	11
1.1.2	OBJETIVOS ESPECÍFICOS	11
2	DEFINIÇÃO DO PROBLEMA	12
3	FUNDAMENTAÇÃO TEÓRICA	15
3.1	PROBLEMA DO CAIXEIRO VIAJANTE	15
3.2	PROBLEMA DO CAIXEIRO VIAJANTE ELÉTRICO	17
3.3	MÉTODOS HEURÍSTICOS	18
3.3.1	Buscas Locais	19
3.3.2	Métodos Construtivos	22
3.3.3	Metaheurísticas	24
3.3.4	Combinação de Heurísticas Construtivas e Metaheurísticas	28
4	TRABALHOS RELACIONADOS	30
4.1	THE ELECTRIC VEHICLE-ROUTING PROBLEM WITH TIME WINDOWS AND RECHARGING STATIONS	30
4.2	THE ELECTRIC TRAVELING SALESMAN PROBLEM WITH TIME WINDOWS	32
4.3	THE ELECTRIC VEHICLE TOURING PROBLEM	34
4.4	COMPARAÇÃO DOS TRABALHOS RELACIONADOS	35
5	METODOLOGIA	37
5.1	TESTES	39
5.2	EXECUÇÃO	40
5.3	MÉTRICAS DE AVALIAÇÃO	40
6	ANÁLISE DOS RESULTADOS	41
6.1	20 CLIENTES	42
6.2	150 CLIENTES	46
6.3	200 CLIENTES	49
7	CONCLUSÕES	53
7.1	TRABALHOS FUTUROS	53
	REFERÊNCIAS	55

1 INTRODUÇÃO

O avanço da tecnologia para carros elétricos vem transformando radicalmente o setor de transportes e logística em todo o mundo. Segundo dados recentes da [International Energy Agency \(2025\)](#) revelam que as vendas globais de veículos elétricos (VEs) ultrapassaram 17 milhões de unidades em 2024, representando mais de 20% do mercado automotivo global. Este crescimento é particularmente significativo quando comparado com períodos anteriores: apenas o incremento de 3,5 milhões de unidades vendidas em 2024 em relação a 2023 supera o total global de veículos elétricos comercializados em todo o ano de 2020.

O primeiro trimestre de 2025 já indica uma aceleração ainda mais pronunciada, com crescimento interanual de 35% nas vendas mundiais, projetando que este ano ultrapassará a marca de 20 milhões de unidades vendidas - o equivalente a mais de um quarto de todos os veículos leves comercializados globalmente.

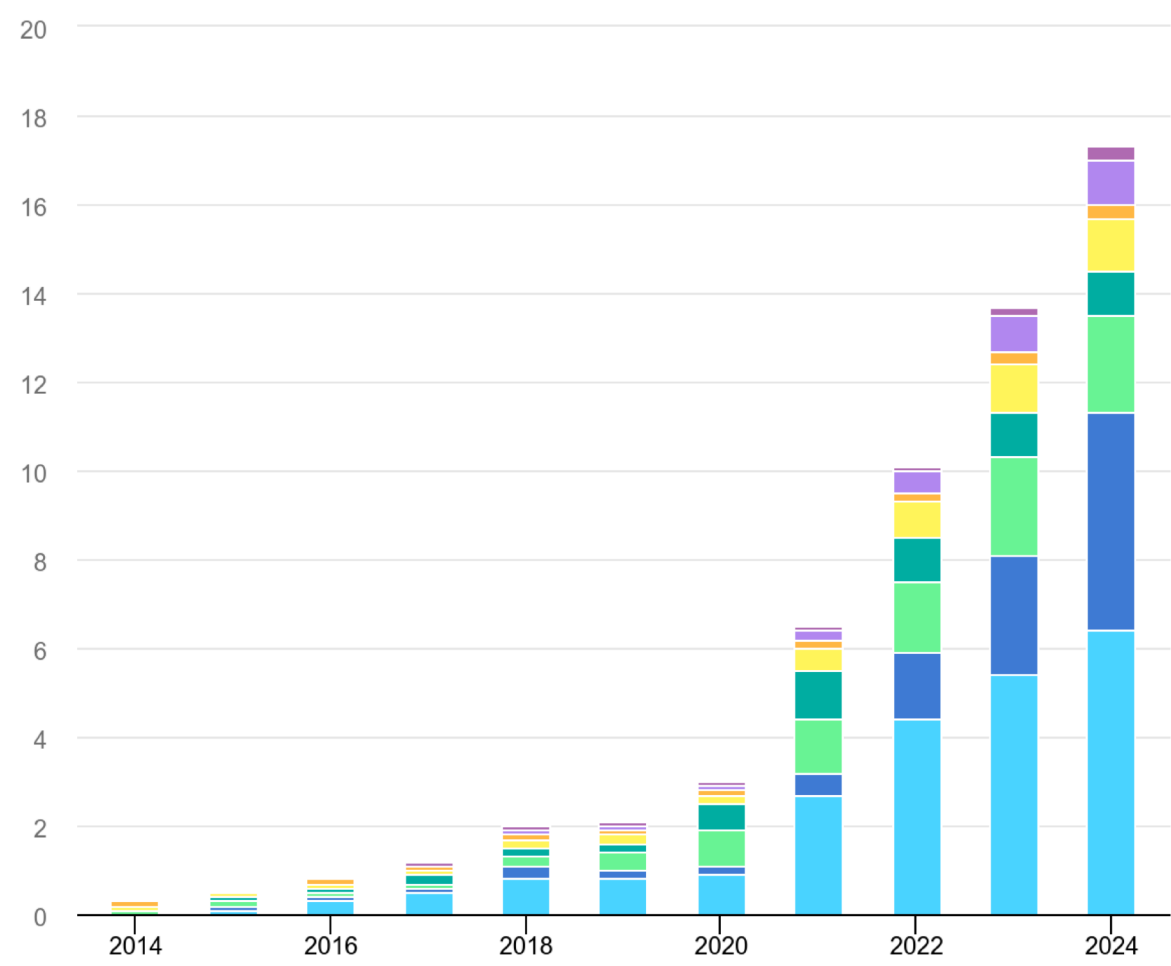


Figura 1 – Vendas globais de veículos elétricos de 2014 a 2024 (em milhões de unidades). Cada cor representa um mercado regional distinto.

Fonte: Gráfico disponível em [International Energy Agency \(2025\)](#).

Esse crescimento na venda dos veículos elétricos pode ser visto na Figura 1, que apresenta

os dados de 2014 a 2024 no eixo horizontal, enquanto o eixo vertical indica o total de veículos vendidos em milhões de unidades, e cada cor representa um dos principais mercados regionais globais.

No contexto logístico, estima-se que até 2030, 30% dos veículos comerciais leves serão elétricos. Essa transição, motivada tanto por pressões ambientais quanto por vantagens econômicas, traz consigo novos desafios computacionais na otimização de rotas e gestão de frotas.

O problema do caixeiro viajante para veículos elétricos (PCV-E) surge como uma das questões centrais nesta nova realidade. Apesar do PCV clássico já existir como um problema amplamente estudado na literatura, diferentemente dele, o PCV-E introduz novas questões críticas relacionadas à autonomia da bateria, disponibilidade de infraestrutura de recarga e padrões de consumo energético variáveis. Esses detalhes adicionais podem aumentar significativamente a complexidade computacional dos problemas de roteamento quando comparados às suas versões convencionais.

Um exemplo concreto das dificuldades operacionais pode ser observado na distribuição urbana de mercadorias com veículos elétricos. Conforme [Soviyan \(2024\)](#), a otimização de rotas para EVs enfrenta desafios significativamente maiores em comparação com veículos convencionais, particularmente devido à restrições de capacidade da bateria nos modelos de roteirização. O estudo revela que essas limitações exigem abordagens especializadas como metaheurísticas híbridas para resolver o "problema de roteirização de veículos elétricos com capacidade". A revisão sistemática identificou que a "distribuição desigual e disponibilidade limitada de estações de carregamento" representam impedimentos críticos, necessitando de algoritmos que considerem não apenas a distância, mas também a localização estratégica de pontos de recarga ao longo das rotas, aumentando substancialmente a complexidade computacional do planejamento logístico.

Considerando que o setor de transportes e os veículos de estradas correspondem por 74% das emissões de CO₂ desse setor relacionadas à energia [IEA \(2023\)](#), mesmo pequenas otimizações nesses percursos podem gerar impactos significativos: uma redução de apenas 1% na distância percorrida por veículos de carga emissores de CO₂ em operações urbanas representaria uma economia anual de aproximadamente 1,2 milhão de toneladas de CO₂ em escala global.

A complexidade inerente ao PCV-E o classifica como um problema NP-difícil. Algoritmos exatos como o *Bellman-Held-Karp* possuem complexidade $O(n^2 \cdot 2^n)$, tornando-se inviáveis para instâncias com mais de algumas dezenas de cidades. Embora *solvers* modernos de programação inteira (como o Gurobi ou o Concorde) consigam resolver instâncias do PCV com centenas de cidades em tempo razoável, a obtenção garantida de soluções ótimas para instâncias reais de grande escala com restrições adicionais — como as impostas pelo PCV-E — permanece desafiadora ([MIYAZAWA, 2021](#)). Nesse contexto, heurísticas bem projetadas conseguem fornecer soluções de alta qualidade em tempo computacional reduzido, tornando-se a abordagem preferencial para problemas de grande escala.

Este trabalho se propõe a desenvolver uma heurística híbrida para o PCV-E, combinando as vantagens do *Variable Neighborhood Search* (VNS) e *Tabu Search* (TS). Nossa abordagem parte de dois pilares fundamentais:

- **Modelagem clássica do PCV:** Através de algoritmos comprovados pela literatura, gerar soluções viáveis e reduzir a distância percorrida total, satisfazendo as restrições de autonomia do problema, será usado métodos de construção e metaheurísticas respectivamente.
- **Otimização em duas fases:** Combinação de busca local intensiva com mecanismos de diversificação, seguindo a estrutura comprovada por [Schneider, Stenger e Goeke \(2014\)](#), simplificando para nosso contexto de apenas um veículo e sem restrições de janela de tempo.

A metodologia proposta será validada através de benchmarks públicos e comparada com outras abordagens da literatura, utilizando como métricas principais: (1) distância total percorrida, (2) tempo computacional, (3) número de recargas necessárias, (4) tamanho de escopo. Como critério de sucesso, almejamos superar em pelo menos 10% os resultados das heurísticas convencionais para instâncias com até 200 pontos de entrega.

A relevância prática desta pesquisa se manifesta em três dimensões principais. Primeiramente, ao reduzir a incerteza operacional associada à adoção de veículos elétricos em frotas comerciais. Em segundo lugar, ao oferecer uma ferramenta computacional acessível para logística. Por fim, ao contribuir para os esforços globais de descarbonização do setor de transportes, conforme estabelecido no Acordo de Paris.

1.1 OBJETIVOS

Esta seção apresenta o objetivo geral e os objetivos específicos deste trabalho.

1.1.1 OBJETIVO GERAL

Especificar uma heurística computacional para lidar com o problema do caixeiro viajante para veículos elétricos.

1.1.2 OBJETIVOS ESPECÍFICOS

Para atingir o objetivo geral, pretende-se cumprir os seguintes objetivos específicos:

- Levantar soluções para o problema de caixeiro viajante para veículos elétricos;
- Definir uma implementação adequada levando em consideração as restrições do problema;
- Definir limites satisfatórios de número de iterações dos algoritmos, equilibrando qualidade da solução e custo computacional;
- Especificar a heurística computacional;
- Analisar resultados dos experimentos sobre a heurística especificada;
- Comparar resultados com trabalhos relacionados da literatura.

2 DEFINIÇÃO DO PROBLEMA

Dado um **grafo** completo dirigido $G = (V, E)$, onde os **vértices** $V = C \cup R \cup \{o\}$, sendo: $C = \{c_1, \dots, c_n\}$ o conjunto de clientes a serem visitados; $R = \{r_1, \dots, r_m\}$ o conjunto de estações de recarga; o o **vértice de origem/destino** (depósito), sem função de recarga — diferentemente de algumas formulações da literatura (ROBERTI; WEN, 2016), neste trabalho o depósito não atua como estação de recarga; E o conjunto de arcos com custo $w((u, v)) \geq 0$ associado ao deslocamento de u a v ;

Deseja-se encontrar uma **rota** $S = \langle v_1, v_2, \dots, v_k \rangle$ que passe por todos os clientes exatamente uma vez, partindo e retornando ao **vértice de origem**, com inserção de estações de recarga conforme necessidade energética, minimizando a função de custo definida a seguir. As estações de recarga em R podem ser visitadas zero ou mais vezes ao longo da rota, e o depósito o aparece como primeiro e último vértice. A rota é, portanto, hamiltoniana em relação ao conjunto de clientes C , mas não necessariamente em relação ao conjunto completo de vértices V , uma vez que estações de recarga podem ser repetidas (SCHNEIDER; STENGER; GOEKE, 2014).

VARIÁVEIS E PARÂMETROS

- $e((u, v)) \geq 0$: consumo de energia no arco $(u, v) \in E$
- $B > 0$: capacidade máxima da bateria (autonomia)
- $b_0 \in [0, B]$: energia inicial no vértice de origem o ; neste trabalho assume-se $b_0 = B$ (bateria cheia no início)
- $b_i \in [0, B]$: nível de energia disponível ao *chegar* no vértice v_i , para $i = 1, \dots, k - 1$ (variável de estado atualizada conforme a equação de recarga abaixo)

FORMULAÇÃO

Seja $S = \langle v_1, v_2, \dots, v_k \rangle$ um ciclo com $v_1 = v_k = o$. A energia b_i é atualizada conforme:

$$b_{i+1} = \begin{cases} B - e((v_i, v_{i+1})) & \text{se } v_i \in R, \\ b_i - e((v_i, v_{i+1})) & \text{caso contrário,} \end{cases} \quad (1)$$

Com o objetivo de minimizar a distância total da rota, considerando:

$$\min \left(\sum_{i=1}^{k-1} w((v_i, v_{i+1})) + \alpha \cdot |\{v_i \in S : v_i \in R\}| \right) \quad (2)$$

onde:

- $w((u, v))$ representa a distância entre dois pontos

- $\alpha \geq 0$ é o custo fixo associado a cada recarga realizada
- $|\{v_i \in S : v_i \in R\}|$ conta o número de estações de recarga visitadas na rota

Essa formulação está sujeita às restrições:

1. **Visita obrigatória:** Todos os clientes $c \in C$ devem ser visitados exatamente uma vez.
2. **Recarga completa e obrigatória:** Todo vértice $v_i \in R$ visitado resulta em recarga completa ($b_i = B$ após recarga).
3. **Bateria não negativa:** A energia b_i ao chegar no vértice v_i deve satisfazer $b_i \geq 0$.
4. **Visita opcional a estações:** As estações de recarga podem ser visitadas ou não, conforme necessidade energética.

ESTRUTURA DO ARQUIVO DE INSTÂNCIA

As instâncias utilizadas seguem um formato padronizado proposto por (ROBERTI; WEN, 2016) sobre o Problema do Caixeiro Viajante Elétrico com Janelas de Tempo (PCV-EJT), organizado em blocos de informações específicos:

O formato das instâncias, tanto do tipo G (20 clientes) quanto OT (150 e 200 clientes), é estruturado da seguinte forma:

```
<number of customers (n)>
<number of recharging stations (m)>
<battery capacity (Q)>
<consumption rate (h)>
<recharging rate (g)>
```

```
*** For the depot, for each customer, and for each recharging station
<id> <x-coordinate> <y-coordinate> <time window start (e)> <time window end (l)>
```

```
<Travel distance/time matrix>
```

O bloco indicado por “***” descreve os dados de cada nó da instância, incluindo o depósito, os clientes e as estações de recarga. Para cada um desses elementos, são informados: um identificador único (id), suas coordenadas no plano cartesiano (x e y) e os limites inferior (e) e superior (l) da janela de tempo associada ao atendimento.

Na sequência, a matriz de distâncias/tempos apresenta os custos de deslocamento entre todos os pares de nós da instância, sendo utilizada para calcular as rotas e avaliar suas respectivas distâncias ou tempos de viagem.

A Figura 2 apresenta uma representação gráfica de uma instância do problema, na qual os nós correspondem ao depósito, aos clientes e às estações de recarga. As posições desses

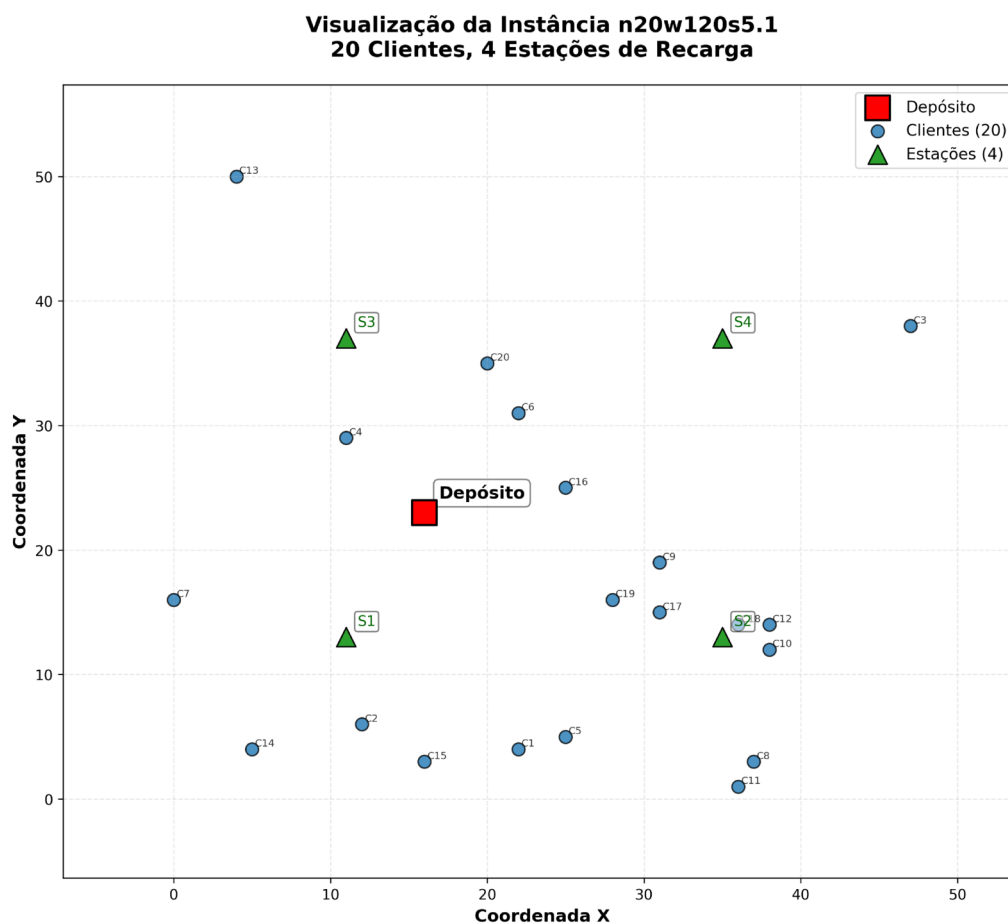


Figura 2 – Representação gráfica de uma instância do ETSP.

elementos são determinadas a partir das coordenadas fornecidas no arquivo da instância, seguindo o padrão estabelecido por (ROBERTI; WEN, 2016).

Cabe destacar que tais instâncias são sintéticas e amplamente utilizadas na literatura para fins de experimentação computacional, permitindo a análise controlada do desempenho de algoritmos em diferentes configurações do problema.

Para esse trabalho que visa resolver o problema do caixeiro viajante para veículos elétricos, iremos desconsiderar as características sobre janelas de tempo das instâncias usadas.

3 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são apresentados os principais conceitos que fundamentam o trabalho. São eles: Caixeiro Viajante e seus algoritmos exatos, métodos construtivos, heurísticas e metaheurísticas.

3.1 PROBLEMA DO CAIXEIRO VIAJANTE

Os problemas de roteamento constituem um campo tradicional de pesquisa em otimização combinatória, sendo que suas versões básicas e variantes são relevantes tanto para investigações acadêmicas quanto para aplicações práticas na indústria. Para estudar variantes como o PCVE, temos que entender inicialmente os clássicos, dentre esses problemas, o Problema do Caixeiro Viajante - PCV (*Traveling Salesman Problem* - TSP) destaca-se como o mais conhecido.

De acordo com [Silveira \(2000\)](#) publicado pela UFRGS o PCV clássico pode ser formalizado da seguinte maneira:

- Um conjunto de clientes que devem ser visitados
- Um veículo único que parte de um depósito inicial
- O veículo deve visitar cada cliente exatamente uma vez
- O veículo deve retornar ao depósito ao final do percurso
- As distâncias entre todos os pares de localizações são conhecidas
- **Objetivo:** Minimizar a distância total percorrida pelo veículo

Matematicamente, o PCV pode ser representado como um problema de otimização em um grafo completo $G = (V, A)$, onde:

- $V = \{v_1, \dots, v_n\}$ é o conjunto de vértices (cidades/clientes)
- A é o conjunto de arcos (conexões entre as cidades)
- Cada arco $(i, j) \in A$ possui um custo associado w_{ij} (distância)
- O objetivo é encontrar o ciclo hamiltoniano de custo mínimo

O PCV é um problema NP-difícil, o que significa que não existem algoritmos conhecidos que possam resolvê-lo em tempo polinomial. Essa característica torna as metaheurísticas particularmente adequadas para sua resolução, especialmente em casos práticos onde soluções aproximadas de boa qualidade são aceitáveis. Porém, isso não quer dizer, que não existam soluções exatas boas para o PCV.

Um exemplo muito famoso da literatura de algoritmo exato para o PCV, é mostrado a seguir.

Algoritmo Bellman-Held-Karp

Destaca-se o algoritmo exato *Bellman-Held-Karp*, demonstrado por [Nguyen \(2020\)](#), um dos métodos mais famosos que resolvem o problema de maneira ótima para instâncias pequenas (abaixo de 20 clientes), conhecido por seu método de programação dinâmica mais eficiente, com complexidade $O(n^2 2^n)$.

Implementação:

- Entrada: Matriz de distâncias D onde $D[i][j]$ é a distância entre os vértices i e j .
- Tabela dp : Armazena o custo mínimo para visitar todos os vértices no conjunto S terminando em j .
- Passo base: Custo para visitar apenas o vértice inicial é zero ($dp[\{1\}][1] = 0$).
- Transição: Para cada subconjunto S e vértice final j , o custo é atualizado combinando soluções de subproblemas menores.
- Solução final: O menor circuito é encontrado adicionando o custo de retorno ao vértice inicial.

O algoritmo calcula sistematicamente os caminhos mínimos para todos os subconjuntos de vértices, construindo a solução final a partir desses subproblemas:

- **Conceito-chave:** Para cada subconjunto S de vértices contendo a origem (vértice 1) e cada vértice final $j \in S$, armazena-se em $dp[S][j]$ o custo do caminho mínimo que:
 - Parte do vértice inicial 1
 - Visita **todos** os vértices em S **exatamente uma vez**
 - Termina no vértice j
- **Construção incremental:** A solução para S é construída combinando soluções de subconjuntos menores $S \setminus j$, adicionando a aresta (i, j) para cada $i \in S \setminus j$.
- **Solução final:** O circuito completo é obtido adicionando o custo de retorno do último vértice visitado à origem.

Algoritmo 1 Algoritmo de Bellman Held-Karp para PCV**Require:** Matriz de distâncias $D_{n \times n}$, vértice inicial $s = 1$ **Ensure:** Custo mínimo do circuito hamiltoniano

```

1: Inicialize uma tabela  $dp[2^n][n]$  com  $\infty$ 
2:  $dp[1][1] \leftarrow 0$  ▷ Custo para visitar apenas o vértice inicial
3: for  $m = 2$  to  $n$  do ▷ Tamanho do subconjunto
4:   for all subconjuntos  $S \subseteq \{1, 2, \dots, n\}$  com  $|S| = m$  e  $1 \in S$  do
5:     for all  $j \in S \setminus \{1\}$  do
6:        $dp[S][j] \leftarrow \min_{\substack{i \in S \\ i \neq j}} (dp[S \setminus \{j\}][i] + D[i][j])$ 
7:     end for
8:   end for
9: end for
10: Calcule o custo final adicionando o retorno ao vértice inicial:
11: return  $\min_{j=2}^n (dp[\{1, 2, \dots, n\}][j] + D[j][1])$ 

```

Este algoritmo representa uma vantagem entre compromisso conhecido entre exatidão e eficiência computacional para instâncias pequenas do PCV, retornando a solução exata de forma mais eficaz que força bruta $O(n!)$, combinando estratégias como a programação dinâmica, dividindo o problema em sub divisões, e evitando calculos redundantes. Porém, o seu uso de memória continua exponencial $O(n2^n)$, se limita praticamente a $n \leq 25$ cidades, por isso, para problemas maiores, é preferível o uso de outros métodos conhecidos que, apesar de não encontrarem a melhor solução, compensam na complexidade computacional.

3.2 PROBLEMA DO CAIXEIRO VIAJANTE ELÉTRICO

O Problema do Caixeiro Viajante Elétrico - PCVE (*Electric Traveling Salesman Problem*) é uma variante do PCV clássico que incorpora restrições específicas de veículos elétricos, o artigo de [Ceselli e Righini \(2020\)](#) da Universidade de Milan mostra com detalhes suas propriedades e modelos. Suas principais características podem incluir (1) **Autonomia limitada**; (2) **Estações de recarga**; (3) **Tempo de recarga**; (4) **Consumo variável**; e entre outras variáveis que podem ser relevantes à pesquisa.

Matematicamente, o PCVE estende a formulação do PCV tradicional com:

- Um conjunto adicional R de nós representando estações de recarga
- Uma função de consumo e_{ij} para cada arco (i, j)
- Uma capacidade máxima de bateria B
- Restrições que garantem que o veículo nunca fique sem carga entre recargas

Essas características tornam o PCVE significativamente mais complexo que o PCV tradicional, exigindo adaptações nas metaheurísticas clássicas para lidar com as novas variáveis e restrições.

A seguir nas próximas seções, são apresentados os principais conceitos que fundamentam as estratégias para resolver o PCVE.

3.3 MÉTODOS HEURÍSTICOS

Esta seção explora os fundamentos desses métodos, destacando seus princípios, aplicações e potencial para resolver problemas complexos, incentivando tanto a pesquisa quanto a aplicação prática dessas abordagens.

De acordo com [Huang, Lai e Cheng \(2009\)](#) sobre algoritmos heurísticos. Eles são uma abordagem prática ou estratégia simplificada usada para resolver problemas ou tomar decisões de maneira eficiente, especialmente quando uma solução exata é muito complexa. Como visto anteriormente no algoritmo exato *bellman-Held-Karp*, para problemas com um número elevado de vértices (custo computacional muito alto), o algoritmo é extremamente ineficaz, e por isso, métodos heurísticos são preferíveis, apesar de não entregarem a melhor solução, geralmente são criadas soluções viáveis, suficientes para o problema, e essa avaliação depende do contexto que está sendo utilizado.

Ademais, há diversas vantagens no uso de heurísticas, dentre elas: (1) Eficiência Computacional, reduzindo drasticamente o tempo de processamento em problemas NP-difíceis (ex: roteamento de veículos, *scheduling*); (2) Aplicabilidade no mundo real, como por exemplo o algoritmo "vizinho mais próximo" para o Problema do Caixeiro-Viajante, *machine learning*, logística, finanças, entre outras áreas de atuação; (3) Flexibilidade e adaptabilidade, pois pode ser ajustada para diferentes contextos e combinada com outros métodos existentes, entregando resultados satisfatórios para problemas complexos.

Para entender todo espectro de métodos heurísticos e seus algoritmos, é necessário ter conhecimento prévio sobre seus principais conceitos:

- Ótimo Local (*local optimum*):

É a melhor solução em uma vizinhança limitada do espaço de busca, mas pode haver soluções melhores em outras regiões, por isso, não se garante a solução exata do problema.

- Ótimo Global (*global optimum*):

Melhor solução em todo o espaço de busca, tendo como objetivo principal em problemas de otimização.

- Paisagem de Otimização (*fitness landscape*):

É uma representação matemática e visual que mapeia todas as soluções possíveis e seus respectivos valores de qualidade (função objetivo). Esse conceito é fundamental para entender o comportamento de algoritmos de busca e otimização.

Dado um espaço de busca S e uma função objetivo $f : S \rightarrow \mathbb{R}$, a paisagem é o conjunto:

$$\mathcal{L} = \{(s, f(s)) \mid s \in S\}$$

onde:

- S = espaço de soluções (ex: combinações de parâmetros)
- $f(s)$ = "altura" da paisagem (qualidade da solução)

A análise da paisagem de otimização permite:

- Escolher o algoritmo mais adequado (gradiente vs. metaheurísticas).
- Identificar armadilhas como ótimos locais.
- Projetar mecanismos de escape (ex: mutação em Algoritmos Genéticos).

Dentre os diversos algoritmos heurísticos, se destacam algumas opções que podem ser tanto elaboradas diretamente para uma situação específica ou que servem para vários tipos de problemas comuns. Entre as principais categorias, se destacam as buscas locais, métodos construtivos e metaheurísticas.

3.3.1 Buscas Locais

De acordo com o livro *Metaheuristic Search Concepts chapter 3* (ZÄPFEL; BRAUNE; BÖGL, 2009), as buscas locais (*local search* - LS) operam sob o princípio fundamental de modificação iterativa de soluções, visando obter progressivamente soluções melhores. Este conceito, embora frequentemente associado à classe de métodos de busca local, é na verdade, mais abrangente, servindo como base para diversas metaheurísticas.

Por isso, podemos definir que buscas locais são métodos iterativos de otimização que partem de uma solução inicial e exploram soluções vizinhas, movendo-se progressivamente para soluções de melhor qualidade dentro de uma região definida do espaço de busca. Diferentemente de métodos exaustivos, as buscas locais focam em regiões promissoras do espaço de soluções, realizando modificações (*moves*) incrementais na solução atual.

As técnicas de busca local são particularmente úteis para:

- Resolver problemas NP-difíceis de grande escala onde métodos exatos são inviáveis
- Encontrar soluções de boa qualidade em tempo computacional reduzido
- Melhorar soluções obtidas por outros métodos (como heurísticas construtivas)
- Explorar eficientemente espaços de busca complexos e não-lineares

Os métodos de busca local fundamentam-se em dois aspectos essenciais, vizinhanças e modificações.

A noção de vizinhança é central nas buscas locais. Em vizinhanças de soluções, cada modificação deve gerar uma solução que difira minimamente da original, caracterizando uma relação de vizinhança. Formalmente, dada uma solução s , uma solução vizinha $s' \in N(s)$ é obtida aplicando-se um operador de movimento pré-definido (denominados *moves*) através de

pequenas alterações, onde $N(s)$ representa o conjunto de todas as soluções alcançáveis por tais modificações.

E para multiplicidade de modificações, para qualquer solução s , existe tipicamente um conjunto considerável de modificações possíveis, gerando assim uma vizinhança $N(s)$ com múltiplas soluções adjacentes.

Outro conceito fundamental para entender buscas locais são ótimos locais e ótimo global. Este comportamento representa limitações importantes: A qualidade da solução depende completamente do ponto de partida; O algoritmo pode ficar "preso" em soluções subótimas, especialmente em problemas com muitos ótimos locais.

A seguir, são descritos dois exemplos de busca local aplicados ao PCV.

2-opt

Um dos exemplos clássicos é o algoritmo **2-opt** para o Problema do Caixeiro Viajante (PCV), baseado na explicação e visualização do problema mostrado por [Davis \(2022\)](#). O *2-opt* opera através de um princípio fundamental: a modificação local de uma rota existente para obter uma solução melhor. O processo começa com uma rota inicial - que pode ser construída de forma aleatória ou através de outra heurística - e então, realiza melhorias iterativas. A cada passo, o algoritmo seleciona duas arestas não adjacentes na rota atual, desconecta essas duas arestas e reconecta os pontos de forma diferente, criando um novo caminho.

É importante destacar que essas operações sempre mantêm a viabilidade da solução, ou seja, a rota continua sendo um ciclo hamiltoniano que visita cada cidade exatamente uma vez. A mudança só é efetivamente implementada se resultar em uma redução no custo total do percurso. Essa decisão é tomada comparando a soma das distâncias das arestas originais com a soma das novas arestas. Se a nova configuração for mais vantajosa, ela é adotada; caso contrário, o algoritmo mantém a rota como estava e testa outro par de arestas.

Esse comportamento pode ser visualizado a seguir:

Uma das principais qualidades do 2-opt é sua simplicidade conceitual e de implementação. O algoritmo não requer estruturas de dados complexas nem operações matemáticas avançadas. Além disso, ele é capaz de melhorar significativamente soluções iniciais, muitas vezes transformando uma rota construída aleatoriamente em uma solução de qualidade aceitável com um número relativamente pequeno de iterações.

Apesar de suas vantagens, o 2-opt apresenta algumas limitações importantes. A mais notável é sua tendência a ficar preso em ótimos locais - situações em que nenhuma troca simples de arestas consegue melhorar a solução atual, mesmo que existam configurações melhores globalmente. Isso ocorre porque o algoritmo só considera modificações locais e não possui mecanismos para escapar desses pontos de estagnação.

Em termos de eficiência computacional, o 2-opt tem complexidade $O(n^2)$ por iteração, onde n representa o número de cidades. Isso significa que, para instâncias muito grandes do problema, o tempo de execução pode se tornar proibitivo. Em tais casos, é comum combinar

Algoritmo 2 Algoritmo 2-opt para o PCV**Require:** Rota cíclica $T = (t_0, t_1, t_2, \dots, t_{n-1})$, com $t_n \equiv t_0$ (depósito)**Ensure:** Rota melhorada T

```

1: melhorou  $\leftarrow$  VERDADEIRO
2: while melhorou do
3:   melhorou  $\leftarrow$  FALSO
4:   for  $i \leftarrow 0$  to  $n - 2$  do
5:     for  $j \leftarrow i + 2$  to  $n - 1$  do ▷  $t_{j+1}$  usa aritmética modular:  $t_n \equiv t_0$ 
6:        $\Delta \leftarrow d(t_i, t_j) + d(t_{i+1}, t_{(j+1) \bmod n}) - d(t_i, t_{i+1}) - d(t_j, t_{(j+1) \bmod n})$ 
7:       if  $\Delta < 0$  then ▷ Troca reduz o custo
8:         Inverter a sub-rota  $t_{i+1}, t_{i+2}, \dots, t_j$  ▷ Inversão inclui  $t_{i+1}$  e  $t_j$ 
9:         melhorou  $\leftarrow$  VERDADEIRO
10:      end if
11:    end for
12:  end for
13: end while
14: return  $T$ 

```

o 2-opt com outras técnicas ou utilizar implementações otimizadas que reduzam o número de operações necessárias.

k-opt

De acordo com a pesquisa feita por [Heimann, Hoang e Hougardy \(2024\)](#), o **k-opt** é uma generalização do algoritmo 2-opt que permite a remoção e reconexão de até k arestas em uma rota do Problema do Caixeiro Viajante (PCV). Enquanto o 2-opt troca apenas duas arestas por vez, o k-opt examina trocas mais complexas, aumentando as chances de escapar de ótimos locais.

Algoritmo 3 Algoritmo k-opt para PCV**Require:** Rota inicial $T = (t_1, t_2, \dots, t_n)$, valor de k **Ensure:** Rota melhorada T

```

1: melhorou  $\leftarrow$  VERDADEIRO
2: while melhorou do
3:   melhorou  $\leftarrow$  FALSO
4:   for todas combinações de  $k$  arestas  $(e_1, e_2, \dots, e_k)$  em  $T$  do
5:     Remova  $e_1, e_2, \dots, e_k$ , dividindo  $T$  em  $k + 1$  segmentos
6:     for todas possíveis reconexões dos segmentos do
7:        $T' \leftarrow$  nova rota gerada
8:       if  $\text{custo}(T') < \text{custo}(T)$  then
9:          $T \leftarrow T'$ 
10:      melhorou  $\leftarrow$  VERDADEIRO
11:    break
12:  end if
13: end for
14: end for
15: end while
16: return  $T$ 

```

- Complexidade: O número de combinações cresce como $O(n^k)$, tornando implementações como $k > 3$ impraticáveis para instâncias grandes.
- Eficácia: Para $k = 3$ (3-opt), melhora significativamente soluções do 2-opt
- Aplicações: Usado em metaheurísticas como *Lin-Kernighan* e VNS

3.3.2 Métodos Construtivos

Algoritmos heurísticos de construção de solução são métodos determinísticos ou aleatórios simples que geram uma solução viável inicial para o problema. Eles são rápidos, mas geralmente produzem soluções viáveis para problemas complexos de otimização combinatória por meio de regras práticas e eficientes, sacrificando a garantia de optimalidade em prol da velocidade computacional.

Esses algoritmos constroem soluções passo a passo, adicionando componentes iterativamente com base em critérios determinísticos (como escolher sempre o vizinho mais próximo no PCV) ou probabilísticos. Diferentemente das metaheurísticas, que refinam soluções existentes, os métodos construtivos iniciam do zero, sendo particularmente úteis para gerar soluções iniciais que, posteriormente, podem ser melhoradas.

Exemplos clássicos incluem o algoritmo do **vizinho mais próximo**, a **inserção mais barata** e o algoritmo de [Christofides \(1976\)](#), que, apesar de sua simplicidade, frequentemente produz soluções de qualidade razoável para problemas como roteamento de veículos, programação de tarefas e design de redes.

Vizinho Mais Próximo

O Algoritmo do Vizinho Mais Próximo (*Nearest Neighbor* - NN) como demonstrado por ([Rangel-Mondragon \(2012\)](#)) é uma das principais e mais simples heurísticas construtivas e intuitivas para problemas de roteamento. A ideia principal é começar em um nó inicial e, a cada passo, escolher o nó mais próximo ainda não visitado, repetindo até que todos os nós tenham sido incluídos na rota. Seu funcionamento pode ser visto a seguir:

Algoritmo 4 Algoritmo do Vizinho Mais Próximo (PCV)**Require:** Matriz de distâncias $D_{n \times n}$, Nó inicial s **Ensure:** Rota R (ciclo hamiltoniano)

```

1:  $R \leftarrow [s]$                                 ▶ Inicializa a rota com o nó inicial
2:  $V \leftarrow \{1, 2, \dots, n\} \setminus \{s\}$       ▶ Nós não visitados
3:  $u \leftarrow s$                                 ▶ Último nó adicionado à rota
4: while  $V \neq \emptyset$  do
5:    $v \leftarrow \operatorname{argmin}_{j \in V} \{D(u, j)\}$   ▶ Encontra o vizinho mais próximo
6:    $R.append(v)$                                 ▶ Adiciona à rota
7:    $V \leftarrow V \setminus \{v\}$                 ▶ Remove dos não visitados
8:    $u \leftarrow v$                                 ▶ Atualiza o último nó
9: end while
10:  $R.append(s)$                                 ▶ Completa o ciclo (PCV)
11: return  $R$ 

```

O algoritmo tem complexidade $O(n^2)$, pois para cada um dos n nós, é necessário percorrer os nós restantes para encontrar o vizinho mais próximo.

As vantagens de se usar esse algoritmo se dão por sua simplicidade, fácil implementação e rápido para pequenas instâncias, porém, devido à falta de estratégias de otimização, pode gerar soluções distantes da ótima, especialmente em instâncias assimétricas ou com distribuição irregular de nós.

Christofides

O algoritmo aproximado de [Christofides \(1976\)](#) é uma versão que garante soluções de até no máximo 1,5 vezes o custo da solução ótima global. Ele possui essa garantia diferente de uma busca heurística qualquer.

De forma simplificada, ele consiste em três etapas principais: (1) Construção de uma Árvore Geradora Mínima (*Minimum Spanning Tree* - MST), um subgrafo que conecta todos os vértices com o menor custo total possível, sem formar ciclos; (2) Seleção de um emparelhamento perfeito mínimo nos vértices de grau ímpar da MST, para gerar um conjunto de arestas M que conecta todos os vértices ímpares com custo mínimo; (3) Geração de um ciclo euleriano e sua conversão em um ciclo hamiltoniano. Seu funcionamento pode ser visto a seguir:

Algoritmo 5 Algoritmo de Christofides (PCV Métrico)**Require:** Grafo completo $G = (V, E)$ com pesos métricos w **Ensure:** Ciclo hamiltoniano R

```

1:  $T \leftarrow \text{Prim}(G)$                                 ▶ Árvore Geradora Mínima (MST)
2:  $O \leftarrow \{v \in V \mid \text{grau}(v) \text{ em } T \text{ é ímpar}\}$   ▶ Vértices de grau ímpar
3:  $M \leftarrow \text{EmparelhamentoPerfeitoMínimo}(O, G)$   ▶ Emparelhamento em  $O$ 
4:  $H \leftarrow T \cup M$                                 ▶ Grafo euleriano (todos os vértices têm grau par)
5:  $\text{EulerCycle} \leftarrow \text{Hierholzer}(H)$                 ▶ Ciclo euleriano em  $H$ 
6:  $R \leftarrow \text{Shortcut}(\text{EulerCycle})$                 ▶ Remove vértices repetidos (atalhos)
7: return  $R$ 

```

Prim (1957) é utilizado para construir a Árvore Geradora Mínima (MST). Já o ciclo euleriano é encontrado usando o algoritmo de **Hierholzer (1873)**, que eficientemente percorre todas as arestas exatamente uma vez.

O algoritmo tem complexidade $O(n^3)$ devido ao emparelhamento. Suas vantagens incluem boas soluções de até 1,5 vezes a solução ótima global, bom funcionamento em grafos métricos e melhores resultados que heurísticas construtivas simples como o vizinho mais próximo e a inserção mais barata. Porém, perde para essas opções devido à sua implementação complexa e resultados mais demorados.

3.3.3 Metaheurísticas

De acordo com os livros *Metaheuristic Search Concepts* e *Handbook of Metaheuristics* dos autores **Zäpfel, Braune e Bögl (2009)** e **Gendreau e Potvin (2010)** respectivamente.

Metaheurísticas são estratégias de alto nível que orientam métodos heurísticos na busca por soluções eficientes para problemas complexos de otimização. Em sua definição original, são métodos de solução que coordenam a interação entre procedimentos de melhoria local e estratégias de alto nível, criando um processo capaz de escapar de ótimos locais e realizar uma busca robusta no espaço de soluções.

Embora não exista uma definição única na literatura, alguns aspectos são consensuais: (1) **independência do problema**, permitindo adaptação a diversos contextos com poucas modificações; (2) **balanceamento entre intensificação** (exploração de regiões promissoras) e **diversificação** (busca em áreas não exploradas); e (3) **natureza iterativa**, combinando mecanismos como buscas locais, estruturas de vizinhança e aprendizado adaptativo. Essas características tornam as metaheurísticas especialmente úteis em problemas logísticos, como o PCV e suas variantes, incluindo o Problema do Caixeiro Viajante Elétrico PCVE.

Com o tempo, esses métodos passaram a incluir qualquer procedimento que empregue estratégias para superar a armadilha da otimalidade local em espaços de solução complexos, especialmente aqueles que utilizam estruturas de vizinhança para definir movimentos admissíveis entre soluções ou para construí-las e destruí-las em processos iterativos.

Em metaheurísticas, é importante entender os conceitos de *exploration* (exploração) e *exploitation* (exploração), eles representam dois pilares fundamentais para a busca de soluções eficientes. A *exploration* refere-se à diversificação da busca, onde o algoritmo investiga regiões inexploradas do espaço de soluções, visando escapar de ótimos locais e identificar áreas promissoras. Por outro lado, a *exploitation* corresponde à intensificação da busca em regiões já conhecidas, refinando soluções existentes para melhorar sua qualidade. O equilíbrio entre esses dois mecanismos é crucial: excesso de exploração pode levar a uma busca dispersa e ineficiente, enquanto ênfase excessiva na exploração pode resultar em estagnação em soluções subótimas.

A exploração de vizinhanças varia conforme o tipo de metaheurística. Em métodos baseados em população, como os *algoritmos genéticos*, as vizinhanças são implicitamente definidas por regras de troca (como o *crossover*), enquanto em outras abordagens, como o *path*

relinking, estruturas mais amplas são utilizadas, incluindo vizinhanças construtivas e destrutivas. Métodos híbridos que combinam busca local com estratégias evolutivas também fazem uso dessas estruturas de forma mais abrangente. Por outro lado, abordagens de “único fluxo” (como algoritmos de busca local pura) manipulam vizinhanças de maneira diversificada, incorporando desde estratégias aleatórias até determinísticas, dependendo da fase de busca ou do histórico do processo (como em métodos baseados em memória, por exemplo, a *Programação por Memória Adaptativa*).

As metaheurísticas ganharam destaque como abordagens preferenciais para problemas complexos, especialmente os combinatórios, devido à sua eficácia prática. Embora não garantam a otimalidade das soluções encontradas, frequentemente superam métodos exatos em tempo computacional e geram soluções de qualidade para problemas em que os métodos exatos não conseguiriam resolver, principalmente em problemas do mundo real. Além disso, estratégias metaheurísticas têm sido integradas a métodos exatos, melhorando seu desempenho.

Essa área abrange uma variedade de técnicas, incluindo *Variable Neighborhood Search*, *Scatter Search*, *Tabu Search*, *Algoritmos Genéticos*, *Colônia de Formigas*, *Simulated Annealing*, entre outras, cada uma com particularidades que as tornam adequadas a diferentes contextos. A flexibilidade das metaheurísticas permite adaptações conforme as características do problema, experiência prévia e preferências do analista, consolidando-as como ferramentas poderosas para otimização.

Para este documento, vamos focar nas metaheurísticas mais usadas na literatura para o PCVE em específico. É importante estudar seu funcionamento e suas características para compreender os motivos de seu uso.

Busca por Vizinhança Variável

O princípio fundamental do *Variable Neighborhood Search* (VNS) é a alternância entre diferentes estruturas de vizinhança durante o processo de busca. Ao contrário de métodos de busca local tradicionais que usam uma única vizinhança, o VNS explora múltiplas vizinhanças de forma sistemática. Esta abordagem permite: (1) Escapar de ótimos locais através da perturbação da solução corrente; (2) Explorar diferentes regiões do espaço de busca; (3) Combinar estratégias de diversificação e intensificação.

O VNS consiste em três componentes principais:

- Estruturas de Vizinhança:

São definidas k_{max} estruturas de vizinhança diferentes ($\mathcal{N}_1, \mathcal{N}_2, \dots, \mathcal{N}_{k_{max}}$), onde tipicamente as vizinhanças com índices menores correspondem a modificações mais sutis na solução, enquanto as com índices maiores representam perturbações mais significativas.

- Procedimento de Perturbação (*Shaking*):

Este componente é responsável por gerar uma nova solução a partir da vizinhança corrente:

$$s' \in \mathcal{N}_k(s)$$

onde s é a solução corrente e k indica a vizinhança sendo utilizada.

- Busca Local:

Após a perturbação, aplica-se um método de busca local para refinar a solução obtida:

$$s'' = \text{LocalSearch}(s')$$

O algoritmo básico do VNS pode ser descrito da seguinte forma:

Algoritmo 6 Variable Neighborhood Search (VNS)

Require: Espaço de soluções S , conjunto de estruturas de vizinhança $\mathcal{N}_k$ ($k = 1, \dots, k_{\max}$)

Ensure: Melhor solução encontrada s^*

```

1:  $s^* \leftarrow \text{ConstructInitialSolution}()$ 
2: while critério de parada não satisfeito do
3:    $k \leftarrow 1$ 
4:   while  $k \leq k_{\max}$  do
5:      $s' \leftarrow \text{Perturbação}(s^*, \mathcal{N}_k)$  ▷ Fase de diversificação
6:      $s'' \leftarrow \text{BuscaLocal}(s')$  ▷ Fase de intensificação
7:     if  $f(s'') < f(s^*)$  then ▷  $f$  é a função objetivo
8:        $s^* \leftarrow s''$ 
9:        $k \leftarrow 1$  ▷ Reinicia com a menor vizinhança
10:    else
11:       $k \leftarrow k + 1$  ▷ Tenta uma vizinhança maior
12:    end if
13:  end while
14: end while
15: return  $s^*$ 

```

O VNS é uma metaheurística poderosa e flexível para resolver problemas de otimização complexos. Sua capacidade de explorar sistematicamente diferentes vizinhanças permite escapar de ótimos locais e encontrar soluções de alta qualidade. A simplicidade conceitual combinada com a eficácia prática tem tornado o VNS uma ferramenta popular e comum na literatura.

Seu conceito virou base para a existência de diversas variações do VNS básico, incluindo:

- VND (*Variable Neighborhood Descent*): Versão determinística que não usa perturbação aleatória
- GVNS (*General VNS*): Combina VNS com outras metaheurísticas
- RVNS (*Reduced VNS*): Versão simplificada que omite a busca local
- PVNS (*Parallel VNS*): Implementação paralela para acelerar a busca

Busca Tabu

O algoritmo *Tabu Search* (TS) se destaca por utilizar mecanismos de memória adaptativa para evitar ciclos e escapar de ótimos locais. Diferentemente de algoritmos puramente aleatórios, a TS emprega uma estratégia inteligente de exploração do espaço de busca, combinando elementos de busca local com regras que proíbem temporariamente certos movimentos (classificados como "tabus"). O princípio fundamental da Busca Tabu reside em três componentes essenciais:

- Memória de curto prazo: Mantém registro dos movimentos recentes através de uma lista tabu, prevenindo o retorno a soluções já visitadas.
- Critério de aspiração: Permite que movimentos tabus sejam aceitos quando levam a soluções excepcionalmente boas.
- Estratégia de vizinhança: Define como transformar a solução atual em soluções candidatas através de operadores específicos.

Matematicamente, o processo pode ser descrito como:

$$s_{t+1} = \arg \min_{s \in N(s_t) \setminus T} f(s)$$

onde $N(s_t)$ é a vizinhança da solução corrente, T é o conjunto de movimentos tabus, e $f(s)$ é a função objetivo.

E o funcionamento da TS pode ser resumido da seguinte maneira:

- Lista Tabu: A lista tabu armazena características dos movimentos recentes por um número fixo de iterações (*tenure*). E pode ser implementada de várias formas.
 - Atributos proibidos: Armazena características específicas da solução
 - Movimentos reversos: Registra os movimentos inversos aos realizados
 - Tamanho dinâmico: Pode variar conforme o comportamento da busca
- Critério de Aspiração: Permite quebrar as regras tabu em casos excepcionais. Os critérios comuns incluem.
 - Melhoria do melhor global
 - Solução nunca visitada
 - Melhoria significativa em relação à média

Algoritmo 7 Metaheurística *Tabu Search***Require:** Função objetivo f , solução inicial s_0 , tamanho da lista tabu L , critério de parada**Ensure:** Melhor solução encontrada s^*

```

1:  $s \leftarrow s_0$                                 ▶ Solução corrente
2:  $s^* \leftarrow s$                                 ▶ Melhor solução global
3:  $T \leftarrow \emptyset$                             ▶ Lista tabu vazia
4:  $iter \leftarrow 0$                                 ▶ Contador de iterações
5: while critério de parada não satisfeito do
6:    $N(s) \leftarrow \text{GeraVizinhança}(s)$                 ▶ Gera soluções vizinhas
7:    $s' \leftarrow \text{MelhorSoluçãoNãoTabu}(N(s), T)$         ▶ Seleciona a melhor solução não tabu
8:   if  $f(s') < f(s^*)$  then                            ▶ Atualiza melhor solução global
9:      $s^* \leftarrow s'$ 
10:  end if
11:   $T \leftarrow \text{AtualizaListaTabu}(T, s \rightarrow s', L)$     ▶ Atualiza memória de curto prazo
12:   $s \leftarrow s'$ 
13:   $iter \leftarrow iter + 1$ 
14: end while
15: return  $s^*$ 

```

A Busca Tabu permanece como uma das metaheurísticas mais populares devido à sua combinação eficiente de simplicidade conceitual e poder de busca. Seu uso inteligente de memória a diferencia de outras abordagens, permitindo uma exploração sistemática do espaço de soluções. Quando adequadamente parametrizada e adaptada ao problema específico, a TS pode produzir resultados de alta qualidade em tempo computacional razoável. Suas vantagens se dão por conta da eficácia em escapar de ótimos locais, adaptabilidade para diferentes problemas, balanceamento natural entre *exploration* e *exploitation* e o número baixo de parâmetros para se ajustar.

3.3.4 Combinação de Heurísticas Construtivas e Metaheurísticas

Tabela 1 – Comparação: Heurísticas Construtivas vs Metaheurísticas

Característica	Construtivas	Metaheurísticas
Entrada	Instância do PCV-E	Solução inicial + instância do PCV-E
Mecanismo	Regras de construção gulosas	Busca em vizinhança com memória ou diversificação
Exploração	Único caminho de construção	Múltiplas regiões do espaço de soluções
Qualidade	Soluções de qualidade variável	Em geral, alta qualidade
Complexidade	$O(n^2)$ a $O(n^3)$	Depende de iterações e vizinhança; geralmente $O(n^2 \cdot \dots)$

Na Tabela 1, a instância do PCV-E corresponde à entrada do problema: as regras do problema, os subconjuntos de V e a matriz de distâncias $|V| \times |V|$, conforme descrito na Seção 2. Na coluna das metaheurísticas, a solução inicial S é a rota gerada por um algoritmo construtivo, que serve de ponto de partida para o processo de busca.

A integração de métodos heurísticos construtivos e metaheurísticas oferece uma abordagem híbrida poderosa para resolver problemas complexos de otimização, como o PCV e suas variantes

(EPCV, VRPTW, etc.)

Heurísticas construtivas (e.g., Vizinho Mais Próximo, [Christofides \(1976\)](#)) geram soluções iniciais viáveis rapidamente, evitando que metaheurísticas (como Tabu Search ou VNS) comecem de soluções aleatórias de baixa qualidade, reduzindo o tempo total de busca, pois partem de uma base já estruturada.

A sinergia entre as abordagens tende a produzir soluções mais próximas do ótimo global do que qualquer uma isoladamente. E isso também se aplica para cenários reais, como em problemas com dados variáveis (e.g., tráfego em tempo real), heurísticas rápidas podem reconstruir soluções básicas, enquanto metaheurísticas as adaptam incrementalmente.

Como demonstrado por ([CHRISTOFIDES, 1976](#)) e ([SCHNEIDER; STENGER; GOEKE, 2014](#)), a combinação de métodos construtivos e metaheurísticos é essencial para problemas de roteamento com restrições complexas.

4 TRABALHOS RELACIONADOS

Nesta seção, são discutidos os principais trabalhos que embasam a abordagem proposta para o *Electric Traveling Salesman Problem* (ETSP), com ênfase em heurísticas e estratégias de otimização para veículos elétricos.

4.1 THE ELECTRIC VEHICLE-ROUTING PROBLEM WITH TIME WINDOWS AND RE-CHARGING STATIONS

[Schneider, Stenger e Goeke \(2014\)](#) propõem uma solução para o *Electric Vehicle-Routing Problem with Time Windows and Recharging Stations* (E-VRPTW), problema que estende o *Vehicle Routing Problem* (VRP) com restrições de bateria, *time windows* e estações de recarga.

Os autores desenvolveram uma heurística híbrida combinando VNS (*Variable Neighborhood Search*) e TS (*Tabu Search*).

Este método heurístico é capaz de fazer buscas locais em vizinhanças cada vez maiores para explorar com eficiência o espaço de soluções e evitar ficar preso em soluções inferiores. Ela já tem sido aplicada com sucesso e com resultados validados em *benchmarks* a outros problemas de otimização combinatória, entre problemas de VRPTW com um ou vários depósitos.

De forma resumida, neste algoritmo híbrido VNS/TS, o componente VNS é usado principalmente para diversificar a busca através de estruturas de vizinhança variáveis. Em etapas como o de perturbação/diversificação, o VNS funciona da maneira tradicional, porém nas etapas de intensificação e aceitação, os critérios claramente se diferenciam.

Por isso, ao contrário da descida local comumente usada, durante a fase de intensificação é empregado o TS no lugar de uma busca local padrão. O TS gera uma vizinhança aleatória de movimentos — amostrando um subconjunto de $N(s)$ ao invés de construir a vizinhança completa, reduzindo o custo computacional — combinando operações como *StationInRE*, *deslocate*, *swap* e *2-opt*. A lista tabu armazena movimentos proibidos temporariamente, permitindo que o algoritmo escape de ótimos locais ao aceitar movimentos que levam a soluções de qualidade inferior à atual quando nenhuma melhoria está disponível. Dessa forma, o TS atua simultaneamente como mecanismo de intensificação (ao selecionar o melhor movimento da vizinhança) e de diversificação (ao evitar ciclos por meio da lista tabu).

Este artigo continua sendo citado por muitas pesquisas como o estado da arte de heurísticas para o E-VRPTW, sua estratégia de *hybrid metaheuristic VNS/TS* trouxe resultados positivos no benchmark e dados bem satisfatórios para comprovar as vantagens em combinar as duas metaheurísticas.

Relação com este trabalho:

- A heurística híbrida (VNS/TS) inspirou a abordagem deste TCC, ainda que o ETSP não considere *time windows* ou múltiplos veículos, o algoritmo apresenta diversas qualidades

e boa capacidade de escalção para tanto soluções simples e mais complexas.

- A gestão de bateria e a otimização de paradas para recarga são desafios compartilhados, mas aqui tratados em um cenário simplificado (único veículo, sem restrições de tempo).

4.2 THE ELECTRIC TRAVELING SALESMAN PROBLEM WITH TIME WINDOWS

Roberti e Wen (2016) propõem uma solução para o *Electric Traveling Salesman Problem with Time Windows* (E-TSPTW), uma variação do TSP clássico que incorpora restrições de *time windows* e autonomia limitada de bateria em veículos elétricos. Os autores propõem um algoritmo heurístico de três etapas para resolver o problema.

Algoritmo esse, que foi baseado e inspirado na proposta feita por Silva e Urrutia (2010) para o TSPTW, uma solução que pode ser considerada o estado da arte de métodos heurísticos para esse problema.

O algoritmo de três etapas funciona da seguinte maneira. A ideia é gerar várias rotas aleatórias completas (*random hamiltonian*), mesmo que inviáveis no início, ou seja, que não respeitam janelas de tempo ou autonomia de bateria, e em seguida refinar cada rota, em três fases, até chegar a soluções viáveis e de boa qualidade.

- Fase 1: Alcançar viabilidade de Janelas de Tempo.

Esta fase visa modificar um TSP de entrada x para que ele satisfaça as restrições da janela de tempo. O procedimento utiliza uma estrutura *Variable Neighborhood Descent* (VND). Para corrigir rotas inviáveis aplicando três operações em sequência: (1) realocar clientes para trás na rota, (2) realocá-los para frente e (3) trocar pares de clientes. Os movimentos são repetidos até que a rota atenda a todas as restrições de *time windows* ou até que nenhuma melhoria adicional seja possível.

- Fase 2: Busca Local

Nesta fase, aplica-se uma busca local baseada em VND para reduzir a distância percorrida em uma rota viável x (que já satisfaz as restrições de *time windows*). Exploram-se iterativamente quatro vizinhanças: *Relocate backward*, *Relocate forward*, *2-opt* e *Swap*.

Em cada vizinhança, executa-se o movimento que oferece a maior melhoria na distância, sem violar as janelas de tempo. O processo termina quando nenhuma melhoria é encontrada em todas as vizinhanças

- Fase 3: Alcançar viabilidade de Capacidade de Bateria

Uma tentativa de obter uma solução viável é realizada através da execução de um algoritmo de Programação Dinâmica.

O objetivo é inserir paradas ótimas em estações de recarga ao longo da rota, garantindo que o nível da bateria permaneça entre 0 e Q durante todo o percurso. Dada uma solução inicial x o procedimento utiliza um algoritmo de rotulagem mono-direcional (*forward labeling*). E retorna a solução y de menor custo que respeite as restrições de bateria, ou indica inviabilidade caso não exista tal solução.

Foram consideradas duas variantes do problema quanto à política de recarga: recarga completa da bateria a cada parada, ou recarga apenas da quantidade necessária para alcançar o

próximo destino. Para cada variante, o algoritmo de três fases foi adaptado, resultando em dois métodos distintos que foram avaliados nos experimentos. Nos testes, ele resolveu rapidamente problemas com 20 clientes (em 0,1 segundo) e também funcionou bem para casos maiores com até 200 clientes, comprovando sua versatilidade e desempenho em diferentes escalas.

4.3 THE ELECTRIC VEHICLE TOURING PROBLEM

Liao, Lu e Shen (2016) propõem algoritmos de aproximação com fator constante para o problema de tour de EVs, que é uma generalização do problema do caixeiro viajante.

Assim como o *Traveling Salesman Problem* (TSP), o *EV Touring Problem* envolve organizar um circuito por várias cidades para minimizar o custo total, visitando cada cidade exatamente uma vez e retornando ao ponto de partida. A diferença fundamental em relação ao TSP clássico está na restrição de autonomia: o veículo elétrico possui capacidade de bateria limitada e precisa realizar paradas em estações de recarga (ou troca de bateria) ao longo do percurso, o que pode exigir desvios do caminho ótimo e introduz um grau adicional de complexidade ao problema.

Se for permitido um veículo elétrico recarregar parcialmente em uma estação de recarga ao invés de fazer a troca de baterias. É possível mostrar que o *charging station problem* (CSP) é na verdade, o mesmo do modelo *gas station problem* (GSP) proposto por Khuller, Malekian e Mestre (2011). A diferença sutil entre os modelos é apenas um fator constante.

Por isso, foram propostos dois algoritmos de aproximação dentro de um fator $9/4$ e um fator $9/2$, respectivamente, para o custo uniforme e não uniforme na estação local para o *EV touring problem*. Mas se a bateria possuir uma capacidade suficientemente alta, a proporção de aproximação é a mesma do bem conhecido algoritmo de Christofides para o TSP.

Tendo em vista que o (*EV touring problem*) pode ser facilmente reconhecido como NP-difícil através de uma redução do TSP: basta definir a capacidade da bateria U como a soma dos pesos de todas as arestas (distâncias); assim, o veículo elétrico pode viajar sem precisar trocar a bateria.

E com isso, foi proposto um algoritmo de aproximação para o *Open-path Electric Vehicle Touring Problem* (OEVTP) utilizando abordagens da teoria dos grafos para resolver o problema de roteamento com trajeto fixo. Também foi desenvolvido um algoritmo exato do tipo *branch-and-bound* para o OEVTP e realizados estudos numéricos para comparar o desempenho do algoritmo de aproximação com a metodologia de solução exata. Os resultados computacionais demonstraram a eficácia do algoritmo de aproximação.

4.4 COMPARAÇÃO DOS TRABALHOS RELACIONADOS

Os três artigos analisados abordam problemas de roteamento para veículos elétricos com diferentes enfoques e complexidades. A tabela a seguir apresenta uma comparação sistemática entre eles.

Tabela 2 – Comparação entre os artigos relacionados

Aspecto	Schneider, Stenger e Goeke (2014)	Roberti e Wen (2016)	Liao, Lu e Shen (2016)
Problema	E-VRPTW	E-TSPTW	EV Touring Problem
Nº de veículos	Múltiplos	Único	Único
Restrições	TW, Bateria, Recarga	TW, Bateria	Bateria
Técnica	VNS/TS híbrido	VND + Prog. Dinâmica	Algoritmos Aproximação
Instâncias	Até 500 clientes	Até 200 clientes	Teórico/NP-difícil
Bateria	Recarga parcial	Recarga total/parcial	Troca/recarga
Contribuição	Metaheurística híbrida	Algoritmo em 3 fases	Fator de aproximação

Enquanto Schneider, Stenger e Goeke (2014) aborda um problema de múltiplos veículos com janelas de tempo (E-VRPTW), Roberti e Wen (2016) e Liao, Lu e Shen (2016) focam em problemas de único veículo, com ênfases diferentes (E-TSPTW e *EV Touring Problem*, respectivamente). Apesar de abordarem problemas diferentes, os três trabalhos compartilham várias características fundamentais para o E-TSP. Como a importância de gestão de bateria e uso de metaheurísticas para otimização de rotas.

Cada artigo propõe estratégias diferentes para lidar com a autonomia limitada, seja através de recarga parcial (Schneider, Stenger e Goeke (2014)), programação dinâmica (Roberti e Wen (2016)) ou modelos de troca (Liao, Lu e Shen (2016)).

E sobre o uso de metaheurísticas, embora com abordagens distintas, todos empregam técnicas de otimização combinatória para lidar com o espaço de soluções. Os trabalhos de Schneider e Roberti são mais aplicados, com testes em instâncias reais, enquanto Liao oferece uma abordagem mais teórica com limites de aproximação, com base no *gas station problem* e suas diferenças para veículos elétricos.

Abordagem técnica:

- Schneider, Stenger e Goeke (2014) combina VNS e TS para diversificação e intensificação
- Roberti e Wen (2016) utiliza VND com programação dinâmica em fases
- Liao, Lu e Shen (2016) emprega teoria de aproximação com garantias teóricas

Seleção da Abordagem para este Trabalho

Para o problema ETSP abordado neste TCC, optou-se por basear a solução principal na metaheurística híbrida VNS/TS de Schneider, devido a:

- Adaptabilidade: A estrutura híbrida permite ajustes para simplificar o problema (removendo *Time Windows* - TW e múltiplos veículos)
- Eficiência comprovada: Resultados sólidos em problemas mais complexos sugerem bom desempenho em versões simplificadas
- Balanceamento: Combina diversificação (VNS) com intensificação (TS) de forma equilibrada

Elementos dos outros artigos foram incorporados como complementos:

- Da abordagem de [Roberti e Wen \(2016\)](#), adota-se a noção de fases para viabilidade de bateria, além de ser um exemplo importante para comparação com o método desenvolvido por este trabalho.
- De [Liao, Lu e Shen \(2016\)](#), consideram-se os *insights* sobre limites teóricos de desempenho, pois é o artigo que mais se assemelha com este trabalho, pois não aborda múltiplos veículos e TW.

Esta combinação permite uma solução prática para o ETSP, mantendo rigor teórico e eficiência computacional.

5 METODOLOGIA

Com base nos estudos realizados sobre o Problema do Caixeiro Viajante, e seus diversos métodos de resolução estudados e apresentados pela literatura, é possível compreender muitas das dificuldades, vantagens e problemas encontrados sobre cada tipo de resolução.

Tais características, continuam persistentes em problemas mais modernos, porém, a adição de veículos elétricos traz consigo novas dificuldades. Por isso, é de grande importância a inovação de novos métodos ou adaptações específicas para este problema.

Neste projeto foi feita uma adaptação com base nas estratégias heurísticas demonstradas por [Schneider, Stenger e Goeke \(2014\)](#) modificando-as para atender aos critérios específicos estabelecidos por esse documento.

Foi necessário realizar a implementação de uma metaheurística híbrida combinando os algoritmos *Variable Neighborhood Search* (VNS) e *Tabu Search* (TS), o uso de ambos os métodos é capaz de buscar soluções eficientes e escapar de ótimos locais, aproveitando um grande espaço de soluções.

UM ALGORITMO HÍBRIDO DE DUAS FASES

Nesta seção será descrito uma metaheurística híbrida para o PCV-E. O algoritmo é baseado nas heurísticas propostas por [Schneider, Stenger e Goeke \(2014\)](#) para o *Electric Vehicle Routing Problem (EVRP)*, o qual pode ser considerado o método metaheurístico estado da arte para o problema.

Algoritmo 8 Metaheurística híbrida VNS/TS

Require: Grafo G , bateria B , consumo $e(u, v)$, solução inicial S_0 (vizinho mais próximo)

Ensure: Ciclo S^* factível com custo mínimo

```
1:  $S^* \leftarrow S_0, TabuList \leftarrow \emptyset, k_{max} \leftarrow 6$                                 ▶ k: intensidade da perturbação VNS
2:  $IterMax \leftarrow 100, S_{current} \leftarrow S_0$ 
3: for  $iter \leftarrow 1$  to  $IterMax$  do
4:   Fase de Diversificação (VNS):
5:    $k \leftarrow 1$ 
6:    $S' \leftarrow Perturbação\_VNS(S_{current}, k)$                                 ▶ perturba rota via 2-opt/swap/insertion
7:    $S_{current} \leftarrow S'$ 
8:   Fase de Intensificação (TS):
9:    $S'' \leftarrow TabuSearch(S_{current}, TabuList)$                                 ▶ Atualiza movimentos da tabu list
10:  if  $grade(S'') < grade(S^*)$  then
11:     $S^* \leftarrow S''$                                                             ▶ Aceita se melhorar
12:     $k \leftarrow 1$                                                                 ▶ Reinicia VNS após intensificação
13:  else
14:     $k \leftarrow k + 1$                                                             ▶ Aumenta intensidade
15:  end if
16: end for
17: return  $S^*$ 
```

Algoritmo 9 Componente *Tabu Search*

Require: Solução S , lista tabu $TabuList$, número máximo de iterações $IterMax$ **Ensure:** Solução melhorada S^*

```

1:  $S^* \leftarrow S$ 
2:  $iter \leftarrow 0$ 
3:  $nonImprove \leftarrow 0$ 
4: while  $iter < IterMax$  and  $nonImprove < 0.05 \times IterMax$  do
5:    $Neighborhood \leftarrow GerarVizinhos2Opt(S)$ 
6:    $(S', move) \leftarrow BestNonTabuMove(Neighborhood, TabuList)$     ▶ Seleciona melhor
    movimento não tabu da vizinhanca
7:   if  $grade(S') < grade(S^*)$  then
8:      $S^* \leftarrow S'$ 
9:      $nonImprove \leftarrow 0$ 
10:  else
11:     $nonImprove \leftarrow nonImprove + 1$ 
12:  end if
13:   $TabuList \leftarrow TabuList \cup \{move\}$ 
14:   $iter \leftarrow iter + 1$ 
15: end while
16: return  $S^*$ 

```

Algoritmo 10 Componente *Variable Neighborhood Search*

Require: Solução S , intensidade k **Ensure:** Solução perturbada S'

```

1: function PERTURBAÇÃO_VNS( $S, k$ )
2:    $S' \leftarrow S$ 
3:   for  $i \leftarrow 1$  to  $k + 1$  do
4:      $move \leftarrow \text{random}\{\text{DeslocateForward}, \text{DeslocateBackward}, \text{Swap}, \text{StationInsertion}\}$ 
5:      $S' \leftarrow \text{apply\_move}(S', move)$ 
6:   end for
7:   return  $S'$ 
8: end function
9:
10: function LOCAL_SEARCH_STATION_REMOVAL( $S$ )
11:   Remover estações de recarga desnecessárias de  $S$ 
12:   return  $S$  otimizada
13: end function

```

O algoritmo proposto gera uma sequência de ciclos hamiltonianos perturbados baseada na solução inicial recebida como entrada, os quais não necessariamente satisfazem restrições de capacidade de bateria. Cabe ressaltar que, embora o critério de aspiração seja apresentado na fundamentação teórica da *Tabu Search* (Seção 3), esta implementação não contempla tal mecanismo: o componente TS aceita apenas movimentos não proibidos pela lista tabu. Para cada entrada, as seguintes 3 etapas serão executadas: (1) Perturbação da rota, usando a estratégia VNS com diversas vizinhanças de movimentos (*2-opt*, *swap*, *deslocate forward*, *deslocate backward* e *station insertion*) descrito em 10; (2) Busca local de remoção de estações de recarga, para evitar números desnecessários de estações na rota; (3) Busca Tabu TS, o componente de busca local do algoritmo que substitui a descida local (VND) convencional. O TS intensifica a busca ao selecionar o melhor movimento disponível na vizinhança, e ao mesmo tempo diversifica por meio da lista tabu, que impede o retorno a movimentos recentes e permite aceitar soluções de qualidade inferior à solução atual quando nenhuma melhoria está disponível. Para manter eficiência computacional em instâncias grandes, a vizinhança é construída por amostragem aleatória de movimentos *2-opt* ao invés de enumerar todas as $O(n^2)$ possibilidades; o funcionamento detalhado pode ser observado no Algoritmo 9.

5.1 TESTES

A fim de analisar a eficácia do algoritmo, foi necessário realizar a implementação descrita, além da esquematização, implementação e execução de *benchmarks*. Para isso, é necessário escolher um conjunto de grafos de teste, executar o algoritmo para cada entrada deste conjunto e avaliar os resultados a partir de critérios bem estabelecidos.

O conjunto de instâncias escolhido, foram as 100 instâncias fornecidas por Roberti e Wen (2016), as mesmas utilizadas para os testes do *Electric Traveling Salesman Problem with Time Windows ETSPW*, apesar do problema desse artigo ser diferente do apresentado por este trabalho, as instâncias são perfeitas para os testes, elas combinam valores coesos com a realidade de veículos elétricos e um número elevado de diferentes testes para diversos cenários, por isso, será apenas ignorados as partes relacionadas a janelas de tempo, o que não impactaria no resultados do *PCV – E*, e por fim, ao utilizar as mesmas instâncias, podemos comparar dados que não devem diferir muito desse problema, como por exemplo, tempo computacional e distância total.

Um arquivo auxiliar será criado para testar as 100 instâncias de forma paralela e coletar os dados principais: rota inicial; rota final; distância total de ambas rotas; nota de ambas rotas; tempo de execução; porcentagem de melhora de distância (%) e entre outros dados.

O mesmo será feito para algoritmos de referência, com o intuito de comparar os diferentes tipos de resolução e indicar detalhadamente as vantagens e desvantagens de cada método. Os métodos comparados são: a heurística construtiva *Nearest Neighbor* (NN), a busca local determinística *Variable Neighborhood Descent* (VND) e a metaheurística *Variable Neighborhood Search* (VNS).

5.2 EXECUÇÃO

Para realização dos testes, todos os algoritmos foram testados 3 vezes para todas as instâncias, numa mesma linguagem de programação, *Python 3.10*. O código-fonte está disponível publicamente em <https://github.com/LeonardoSeishi/TCC>. A fim de realizar uma comparação justa do tempo computacional para cada método, foi utilizado um computador Intel(R) Xeon(R) CPU @ 3.60GHz utilizando 8 threads com 32Gb de memória RAM como base. E por último, foi usado dois algoritmos construtivos: (1) Algoritmo do vizinho mais próximo (*Nearest Neighbor NN*); e (2) Geração aleatória de rota. Para serem usadas como solução inicial, com isso, será possível observar tanto a capacidade do algoritmo de melhorar soluções e uma comparação justa entre os algoritmos utilizando o mesmo ponto de partida.

5.3 MÉTRICAS DE AVALIAÇÃO

Os ciclos hamiltonianos gerados pelos algoritmos foram avaliados com base em três principais métricas: (1) **tempo de execução** em segundos; (2) **qualidade de solução**, distância final, garantindo zero violações de bateria. (3) **melhoria de solução**, o quanto o algoritmo foi capaz de diminuir a distância total da solução inicial.

6 ANÁLISE DOS RESULTADOS

As instâncias usadas foram coletadas do trabalho proposto por [Roberti e Wen \(2016\)](#), são 100 instâncias que incluem problemas de 20, 150 e 200 clientes, metade contendo 4 estações e a outra metade 9 estações de recarga, da totalidade de instâncias estão divididas em: 50 problemas com 20 clientes; 30 problemas de 150 clientes; 20 problemas de 200 clientes.

Nos problemas de 4 estações, a organização de clientes e suas posições são idênticas a de 9 estações, apenas com a adição de 5 estações extras, para testar a eficácia do algoritmo em escolher os melhores pontos de recarga na rota.

Este capítulo está dividido em 3 seções, e para cada uma, mais 3 subseções separadas em testes utilizando uma entrada de solução inicial gerada por *nearest neighbor*, e outra com uma solução inicial completamente aleatória e por fim uma análise dos resultados obtidos. Todas as tabelas possuem comparativos entre 4 algoritmos: (1) *Solução inicial (S)*, o qual não garante viabilidade de capacidade de bateria, variando entre *NN* e *Random*; (2) *Variable neighborhood descent (VND)*, heurística simples de busca local; (3) *Variable neighborhood Search (VNS)*, metaheurística, que utiliza o *VND* durante a etapa de intensificação. (4) Metaheurística híbrida (*VNS/TS*), abordada neste trabalho;

Todos os resultados obtidos pelos algoritmos *VND*, *VNS* e *VNS/TS*, garantem a viabilidade da rota por capacidade de bateria.

Os dados relevantes para essa pesquisa serão a distância final D o tempo computacional t em segundos, a média μ desses dados, e a melhoria de solução em relação a D inicial e D final (%), resultados sobre melhoria de solução serão considerados apenas as tabelas de *NN*, tais quais garantem uma comparação justa, possuindo o mesmo ponto de partida para todos os algoritmos.

Por fim, foi comparado esses resultados aos obtidos no artigo de [Roberti e Wen \(2016\)](#) sobre o algoritmo de 3 etapas com recargas completas descrito no trabalho, porém, devido as diferenças de implementação, adição de janelas de tempo e ambiente de testes, apenas dados em relação a distância final foram consideradas, outros fatores como tempo computacional e melhoria de solução foram ignorados, devido a falta de dados sobre a solução inicial utilizada e a diferença entre poder computacional.

No artigo de [Roberti e Wen \(2016\)](#) não existem dados sobre a distância final, apenas do custo final da rota. Considerando as premissas das instâncias - onde 1 unidade de distância equivale a 1 unidade de tempo e consome 1 unidade de energia, e que 0,25 de unidade de tempo recarrega 1 unidade de energia - o custo total incorpora tanto o tempo de viagem quanto o tempo de recarga. Dado que o custo representa a distância acrescida de 25% para recarga ($\text{custo} = D \times 1,25$), a distância D foi calculada como $D = \text{custo} \div 1,25$, permitindo comparação com os resultados dos algoritmos desenvolvidos.

6.1 20 CLIENTES

Tabela 4 – 3P-Heu para instâncias com 20 clientes e 4 estações [Roberti e Wen \(2016\)](#)

Instância	custo	D
n20w120s5.1	271	216.8
n20w120s5.2	233	186.4
n20w120s5.3	317	253.6
n20w120s5.4	314	251.2
n20w120s5.5	249	199.2
n20w140s5.1	181	144.8
n20w140s5.2	279	223.2
n20w140s5.3	238	190.4
n20w140s5.4	265	212.0
n20w140s5.5	229	183.2
n20w160s5.1	246	196.8
n20w160s5.2	219	175.2
n20w160s5.3	210	168.0
n20w160s5.4	208	166.4
n20w160s5.5	253	202.4
n20w180s5.1	262	209.6
n20w180s5.2	273	218.4
n20w180s5.3	282	225.6
n20w180s5.4	206	164.8
n20w180s5.5	201	160.8
n20w200s5.1	241	192.8
n20w200s5.2	221	176.8
n20w200s5.3	255	204.0
n20w200s5.4	296	236.8
n20w200s5.5	240	192.0
Média Geral	247.6	198.1

Tabela 5 – 3P-Heu para instâncias com 20 clientes e 9 estações [Roberti e Wen \(2016\)](#)

Instância	custo	D
n20w120s10.1	270	216.0
n20w120s10.2	222	177.6
n20w120s10.3	312	249.6
n20w120s10.4	308	246.4
n20w120s10.5	243	194.4
n20w140s10.1	179	143.2
n20w140s10.2	277	221.6
n20w140s10.3	237	189.6
n20w140s10.4	260	208.0
n20w140s10.5	225	180.0
n20w160s10.1	245	196.0
n20w160s10.2	208	166.4
n20w160s10.3	210	168.0
n20w160s10.4	208	166.4
n20w160s10.5	248	198.4
n20w180s10.1	254	203.2
n20w180s10.2	272	217.6
n20w180s10.3	273	218.4
n20w180s10.4	206	164.8
n20w180s10.5	199	159.2
n20w200s10.1	239	191.2
n20w200s10.2	213	170.4
n20w200s10.3	250	200.0
n20w200s10.4	295	236.0
n20w200s10.5	233	186.4
Média Geral	243.4	194.7

Solução inicial NN

Tabela 6 – Resultados de distâncias para instâncias de 20 clientes e 9 estações com solução inicial NN

Instância	VND			VNS			VNS/TS			NN μ D
	Melhor D	μ D	(%)	Melhor D	μ D	(%)	Melhor D	μ D	(%)	
n20w120s10.1	195	214.33	10.32	200	201.00	16.74	183	192.00	19.67	239
n20w120s10.2	180	193.33	17.02	184	191.50	15.74	161	162.67	30.19	233
n20w120s10.3	223	229.00	12.60	208	235.00	11.83	188	193.00	26.34	262
n20w120s10.4	201	209.00	22.59	205	211.50	24.20	186	193.00	28.52	270
n20w120s10.5	197	199.67	7.99	197	197.00	9.22	196	196.33	9.52	217
n20w140s10.1	162	166.00	8.29	165	165.00	8.84	162	162.00	10.50	181
n20w140s10.2	197	197.00	21.83	197	209.00	15.48	177	182.00	27.78	252
n20w140s10.3	214	215.67	2.41	215	217.00	1.51	191	191.00	13.57	221
n20w140s10.4	190	190.67	23.43	197	202.00	20.35	162	162.33	34.81	249
n20w140s10.5	179	179.00	5.79	179	179.00	5.79	171	172.33	9.30	190
n20w160s10.1	186	193.33	15.94	201	203.50	11.88	171	181.67	21.01	230
n20w160s10.2	182	184.67	-0.36	182	182.00	1.09	167	172.67	6.16	184
n20w160s10.3	202	206.67	20.82	200	201.50	22.99	175	187.67	28.10	261
n20w160s10.4	167	167.33	20.32	167	167.00	20.79	149	159.67	23.97	210
n20w160s10.5	198	216.00	12.90	203	203.00	18.82	174	179.00	27.82	248
n20w180s10.1	193	200.67	15.33	183	191.00	18.00	174	175.00	26.16	237
n20w180s10.2	203	207.33	7.03	191	199.50	9.72	178	185.33	16.89	223
n20w180s10.3	192	200.67	17.76	202	202.00	17.21	186	187.67	23.09	244
n20w180s10.4	198	198.67	4.94	198	198.00	5.26	173	181.33	13.24	209
n20w180s10.5	162	163.00	26.58	164	165.50	25.53	148	154.67	30.33	222
n20w200s10.1	195	198.33	26.00	190	191.00	26.99	193	195.00	27.24	268
n20w200s10.2	190	199.67	10.86	190	203.00	8.04	178	180.67	19.35	224
n20w200s10.3	192	206.67	4.76	192	197.50	9.37	166	184.00	15.21	217
n20w200s10.4	217	219.00	15.12	220	220.00	15.12	207	209.67	18.73	258
n20w200s10.5	231	234.33	10.56	223	225.50	12.85	210	213.67	18.45	262
Média Geral		199.60	13.06		198.32	14.54		182.17334	21.67	232.44

Tabela 7 – Resultados de distâncias para instâncias de 20 clientes e 4 estações com solução inicial *NN*

Instância	VND			VNS			VNS/TS			NN μ D
	Melhor D	μ D	(%)	Melhor D	μ D	(%)	Melhor D	μ D	(%)	
n20w120s5.1	186	186.00	23.77	187	201.50	17.62	186	201.00	17.62	244
n20w120s5.2	181	202.00	13.30	186	196.00	13.45	166	169.67	27.18	233
n20w120s5.3	199	211.67	19.21	204	216.00	16.29	172	192.67	26.46	262
n20w120s5.4	202	208.33	23.69	193	193.50	26.50	194	194.67	28.69	273
n20w120s5.5	187	187.33	23.85	188	192.50	19.92	190	194.67	20.87	246
n20w140s5.1	173	176.33	3.64	167	170.00	7.65	153	161.67	11.66	183
n20w140s5.2	191	201.00	23.57	196	203.00	23.32	188	190.00	27.76	263
n20w140s5.3	198	205.67	11.73	221	224.50	7.44	193	194.67	16.45	233
n20w140s5.4	186	198.00	22.35	187	194.00	24.84	161	169.67	33.46	255
n20w140s5.5	179	179.00	5.79	179	179.00	5.79	158	172.00	9.47	190
n20w160s5.1	186	197.00	14.35	195	200.50	14.93	168	172.33	25.07	230
n20w160s5.2	173	187.00	-2.19	173	173.00	5.46	164	167.33	8.56	183
n20w160s5.3	200	200.00	23.37	200	205.00	22.09	175	179.33	31.29	261
n20w160s5.4	167	167.00	20.48	165	166.00	20.79	165	165.00	21.43	210
n20w160s5.5	196	202.33	14.63	196	197.00	13.50	174	181.67	23.35	237
n20w180s5.1	195	197.00	17.23	198	201.00	15.69	170	171.33	28.01	238
n20w180s5.2	193	193.00	13.45	193	193.00	13.45	198	198.00	11.21	223
n20w180s5.3	200	201.00	17.62	200	200.00	18.03	182	185.67	23.91	244
n20w180s5.4	195	196.67	8.95	195	195.00	9.57	181	184.00	14.81	216
n20w180s5.5	165	166.67	26.90	170	172.00	26.17	155	159.33	30.12	228
n20w200s5.1	192	204.33	24.88	197	200.50	26.96	197	198.33	27.08	272
n20w200s5.2	187	191.00	14.73	189	198.00	12.05	176	183.33	18.15	224
n20w200s5.3	192	199.33	8.14	203	214.50	3.99	170	173.67	19.97	217
n20w200s5.4	225	228.33	12.52	222	223.50	14.18	216	217.00	16.86	261
n20w200s5.5	215	223.67	14.63	216	217.00	15.90	208	214.33	18.19	262
Média Geral		196.38	15.54		197.04	15.33		183.653336	20.59	235.52

Tabela 8 – Resultados de tempos para instâncias de 20 clientes e 9 estações com solução inicial *NN*

Instância	VND	VNS μ time (s)	VNS/TS	NN time (s)
n20w120s10.1	0.40	5.89	31.84	0.003
n20w120s10.2	0.61	6.27	45.16	0.014
n20w120s10.3	0.80	6.31	29.25	0.002
n20w120s10.4	0.85	6.23	41.27	0.002
n20w120s10.5	0.61	6.19	24.43	0.007
n20w140s10.1	0.54	5.72	17.93	0.015
n20w140s10.2	0.93	6.30	32.88	0.003
n20w140s10.3	0.53	6.39	26.97	0.006
n20w140s10.4	0.74	6.10	45.66	0.027
n20w140s10.5	0.59	6.12	25.34	0.010
n20w160s10.1	0.60	5.87	30.91	0.008
n20w160s10.2	0.51	6.40	40.08	0.021
n20w160s10.3	0.93	6.55	28.54	0.007
n20w160s10.4	0.73	5.99	25.90	0.019
n20w160s10.5	0.44	6.49	28.33	0.013
n20w180s10.1	0.57	6.56	34.08	0.004
n20w180s10.2	0.44	6.16	30.17	0.011
n20w180s10.3	0.56	6.11	22.61	0.012
n20w180s10.4	0.67	6.50	36.64	0.013
n20w180s10.5	0.69	6.10	46.95	0.006
n20w200s10.1	0.95	6.12	39.37	0.003
n20w200s10.2	0.87	6.46	39.71	0.003
n20w200s10.3	0.62	6.16	32.15	0.004
n20w200s10.4	0.64	6.05	21.08	0.003
n20w200s10.5	0.43	6.39	29.03	0.008
Média	0.65	6.22	32.25	0.009

Nota: Tempo médio *tabu Search* do VNS/TS: 0.53sTabela 9 – Resultados de tempos para instâncias de 20 clientes e 4 estações com solução inicial *NN*

Instância	VND	VNS μ time (s)	VNS/TS	NN time (s)
n20w120s5.1	0.56	6.36	23.06	0.003
n20w120s5.2	0.62	5.91	34.71	0.003
n20w120s5.3	0.87	5.89	33.23	0.004
n20w120s5.4	0.76	5.67	32.47	0.008
n20w120s5.5	0.70	6.03	22.07	0.002
n20w140s5.1	0.51	5.87	28.25	0.014
n20w140s5.2	0.80	5.69	35.88	0.003
n20w140s5.3	0.62	6.21	33.55	0.014
n20w140s5.4	0.79	5.93	23.29	0.006
n20w140s5.5	0.52	5.81	18.23	0.019
n20w160s5.1	0.49	5.43	31.96	0.025
n20w160s5.2	0.95	6.01	37.79	0.004
n20w160s5.3	0.99	6.34	37.51	0.025
n20w160s5.4	0.63	6.04	20.84	0.003
n20w160s5.5	0.74	5.83	31.90	0.003
n20w180s5.1	0.59	5.99	31.28	0.007
n20w180s5.2	0.73	5.63	21.00	0.005
n20w180s5.3	0.51	5.68	27.65	0.003
n20w180s5.4	0.74	6.18	40.74	0.003
n20w180s5.5	0.58	6.38	36.50	0.004
n20w200s5.1	0.76	5.86	33.69	0.003
n20w200s5.2	0.76	6.12	34.16	0.003
n20w200s5.3	0.85	5.79	37.47	0.007
n20w200s5.4	0.58	5.81	26.08	0.003
n20w200s5.5	1.03	5.95	38.56	0.005
Média	0.71	5.94	30.87	0.007

Nota: Tempo médio *tabu Search* do VNS/TS: 0.53s

Solução inicial aleatória

Tabela 10 – Resultados de distâncias para instâncias de 20 clientes e 9 estações com solução inicial Random

Instância	VND			VNS			VNS/TS			Random μD
	Melhor D	μD	(%)	Melhor D	μD	(%)	Melhor D	μD	(%)	
n20w120s10.1	197	211.25	65.75	183	185.75	68.47	183	194.25	65.23	586.50
n20w120s10.2	189	194.00	63.50	164	165.75	67.35	164	165.75	69.41	531.33
n20w120s10.3	172	204.25	70.79	171	171.50	74.43	171	177.25	70.69	663.97
n20w120s10.4	186	198.25	77.50	183	189.00	64.73	186	189.50	68.87	8941.28
n20w120s10.5	214	235.75	69.70	178	179.75	79.03	183	190.00	69.37	20047.28
n20w140s10.1	169	184.25	64.46	153	153.00	73.21	161	161.75	70.90	553.33
n20w140s10.2	181	200.25	66.96	180	180.50	68.03	181	185.00	67.88	587.11
n20w140s10.3	207	215.25	67.56	182	185.50	80.00	182	189.50	70.29	11749.31
n20w140s10.4	181	194.00	77.25	158	158.75	75.58	159	162.00	87.89	25646.89
n20w140s10.5	174	188.00	71.49	154	154.75	73.29	154	164.75	79.95	8963.03
n20w160s10.1	182	205.50	83.26	168	170.50	73.38	170	174.00	72.73	17316.61
n20w160s10.2	155	195.75	66.98	150	155.75	75.29	150	154.50	74.05	612.89
n20w160s10.3	185	201.50	70.34	171	171.75	73.42	167	170.50	74.28	669.75
n20w160s10.4	171	187.75	67.39	146	146.50	73.70	154	163.25	78.70	8907.81
n20w160s10.5	205	217.75	64.34	173	174.25	73.15	173	179.00	71.99	636.19
n20w180s10.1	199	206.00	69.13	168	174.50	81.10	168	176.25	81.58	20097.17
n20w180s10.2	191	205.75	76.31	174	177.50	72.57	184	186.75	71.40	8981.06
n20w180s10.3	194	216.75	64.64	182	184.00	70.68	182	186.75	69.71	623.14
n20w180s10.4	195	211.00	65.46	173	175.75	68.52	179	182.75	69.70	594.17
n20w180s10.5	166	177.00	70.52	148	148.25	73.74	152	155.75	73.42	589.61
n20w200s10.1	197	218.50	62.66	190	190.00	69.83	191	196.00	66.15	603.17
n20w200s10.2	185	196.25	64.55	174	174.75	69.67	174	175.75	78.68	8917.50
n20w200s10.3	199	204.50	65.36	167	177.00	68.43	166	180.25	68.10	573.81
n20w200s10.4	219	232.25	66.18	202	205.00	81.30	201	207.50	69.10	11805.83
n20w200s10.5	230	235.00	61.29	202	204.00	65.08	204	209.50	64.16	595.92
Média Geral		205.46	68.53		174.15	72.56		179.13	72.17	6391.79

Tabela 11 – Resultados de distâncias para instâncias de 20 clientes e 4 estações com solução inicial Random

Instância	VND			VNS			VNS/TS			Random μD
	Melhor D	μD	(%)	Melhor D	μD	(%)	Melhor D	μD	(%)	
n20w120s5.1	183	197.75	65.80	183	184.75	66.76	183	191.00	66.09	572.31
n20w120s5.2	184	190.25	73.65	166	167.50	65.63	166	171.50	76.72	17192.00
n20w120s5.3	172	203.00	70.07	172	174.00	75.33	172	182.00	78.36	9010.97
n20w120s5.4	199	219.00	67.16	184	187.00	68.92	193	195.25	69.08	642.39
n20w120s5.5	190	204.75	65.37	187	187.50	66.58	190	195.50	83.55	17253.81
n20w140s5.1	168	172.75	67.46	153	155.00	90.70	153	160.75	69.97	33865.67
n20w140s5.2	189	205.25	65.90	181	187.00	79.69	182	186.25	78.70	20060.75
n20w140s5.3	183	207.75	75.56	183	184.00	98.82	183	188.50	71.51	50639.61
n20w140s5.4	177	190.50	85.72	161	163.25	74.05	161	161.25	74.25	17284.42
n20w140s5.5	179	188.50	70.98	157	159.25	90.30	157	170.50	71.58	22841.50
n20w160s5.1	188	195.75	70.52	168	172.25	72.54	170	175.50	70.60	635.86
n20w160s5.2	159	188.00	69.91	154	154.00	74.96	154	158.50	81.08	8962.83
n20w160s5.3	167	200.00	75.27	167	169.75	90.76	167	169.00	75.95	39565.03
n20w160s5.4	174	186.00	65.56	149	152.00	81.62	149	151.00	72.28	11654.53
n20w160s5.5	200	213.00	67.19	174	176.00	72.12	174	182.50	73.18	661.08
n20w180s5.1	186	195.50	77.21	170	173.75	84.27	170	177.75	80.76	28466.56
n20w180s5.2	204	227.50	65.28	174	179.50	70.79	174	186.50	72.57	658.11
n20w180s5.3	193	223.50	71.50	182	184.00	78.79	182	187.50	68.38	20028.11
n20w180s5.4	182	193.25	84.86	180	180.00	76.96	183	184.75	76.41	36679.00
n20w180s5.5	165	170.25	71.02	151	151.00	73.29	151	155.00	78.93	8912.31
n20w200s5.1	211	221.00	61.68	190	190.75	78.27	195	196.50	66.19	11697.28
n20w200s5.2	187	200.50	74.31	176	178.00	77.95	179	184.25	74.77	28341.11
n20w200s5.3	192	205.50	91.95	171	178.00	71.61	170	183.25	68.48	33932.39
n20w200s5.4	219	223.25	64.06	207	207.50	68.20	207	213.25	69.41	663.86
n20w200s5.5	221	231.75	60.88	208	210.50	61.48	207	208.75	72.00	8910.97
Média Geral		202.17	71.15		176.25	76.42		180.66	73.63	17165.30

Tabela 12 – Resultados de tempos para instâncias de 20 clientes e 9 estações com solução inicial Random

Instância	VND	VNS μ time (s)	VNS/TS
n20w120s10.1	2.66	18.76	72.08
n20w120s10.2	2.47	24.39	101.61
n20w120s10.3	1.66	16.11	70.87
n20w120s10.4	2.02	21.38	97.12
n20w120s10.5	2.02	21.35	79.68
n20w140s10.1	2.53	32.29	79.64
n20w140s10.2	2.93	21.47	95.65
n20w140s10.3	2.61	28.10	80.99
n20w140s10.4	2.23	26.10	100.92
n20w140s10.5	2.75	20.02	93.40
n20w160s10.1	2.24	23.97	88.79
n20w160s10.2	2.63	23.95	77.49
n20w160s10.3	2.21	18.45	107.79
n20w160s10.4	2.27	30.02	72.98
n20w160s10.5	2.68	31.12	107.08
n20w180s10.1	2.08	24.04	78.98
n20w180s10.2	2.03	22.64	78.35
n20w180s10.3	2.23	20.59	69.59
n20w180s10.4	2.40	26.88	131.88
n20w180s10.5	2.37	18.02	87.59
n20w200s10.1	3.45	31.78	108.79
n20w200s10.2	2.31	20.66	108.54
n20w200s10.3	1.90	24.15	67.12
n20w200s10.4	2.23	23.07	83.94
n20w200s10.5	2.59	20.06	66.73
Média	2.38	23.58	88.30

Nota: Tempo médio *tabu Search* do VNS/TS: 0.90s

Tabela 13 – Resultados de tempos para instâncias de 20 clientes e 4 estações com solução inicial Random

Instância	VND	VNS μ time (s)	VNS/TS
n20w120s5.1	1.99	23.19	57.37
n20w120s5.2	2.09	28.59	82.14
n20w120s5.3	2.03	14.77	73.50
n20w120s5.4	2.09	27.78	72.70
n20w120s5.5	2.88	20.81	65.73
n20w140s5.1	2.76	27.52	84.69
n20w140s5.2	2.21	16.95	70.79
n20w140s5.3	2.18	16.14	75.21
n20w140s5.4	2.05	21.60	97.23
n20w140s5.5	2.70	17.24	105.41
n20w160s5.1	2.12	20.50	69.89
n20w160s5.2	2.90	17.81	79.48
n20w160s5.3	2.69	20.61	103.71
n20w160s5.4	1.83	19.81	84.67
n20w160s5.5	2.54	22.35	79.54
n20w180s5.1	2.70	30.27	77.66
n20w180s5.2	2.05	20.43	95.71
n20w180s5.3	2.15	28.69	53.57
n20w180s5.4	2.81	30.93	87.18
n20w180s5.5	2.51	18.84	93.41
n20w200s5.1	2.45	22.74	103.72
n20w200s5.2	2.37	27.41	78.73
n20w200s5.3	2.05	36.79	68.26
n20w200s5.4	1.78	19.43	55.51
n20w200s5.5	2.43	18.54	95.59
Média	2.33	22.79	80.46

Nota: Tempo médio *tabu Search* do VNS/TS: 0.88s

Análise dos resultados

A análise dos resultados apresentados nas Tabelas 6 e 7 permite observar que o algoritmo híbrido VNS/TS obteve melhor desempenho, com uma margem de melhoria de 4% a 7% na distância final das rotas em relação aos demais algoritmos. Esse resultado demonstra a eficácia da *tabu search* em explorar eficientemente o espaço de busca, localizando movimentos ótimos que escapam dos ótimos locais onde as técnicas do VND e VNS convencionais podem ficar presas devido à sua dependência de sequências de movimentos aleatórios.

Embora a diferença de desempenho entre o VND e o VNS seja de aproximadamente 1%, é evidente a vantagem das metaheurísticas em relação às heurísticas simples, graças aos seus mecanismos de diversificação. Vale destacar que, enquanto o VND apresentou alguns resultados superiores ao VNS em instâncias específicas, o VNS demonstrou maior consistência, sem registrar melhorias negativas em nenhuma das execuções. Essa característica pode indicar, contudo, certa ineficiência nos métodos de *shaking* e critérios de parada implementados para o VNS.

Observa-se que as distâncias iniciais obtidas pela heurística do Vizinho Mais Próximo (NN) são, em média, menores nas instâncias com 9 estações de recarga, o que pode explicar a variação reduzida nos resultados entre as diferentes configurações testadas.

A comparação com os resultados das Tabelas 10 e 11, que utilizam soluções aleatórias como ponto de partida, revela três características principais: (1) ambas as metaheurísticas VNS e VNS/TS obtiveram resultados superiores, sugerindo que ambos algoritmos necessitam de

mecanismos de diversificação mais amplos; (2) o *VNS/TS* demonstrou maior eficácia em escapar da solução inicial do *NN*, devido à combinação de dois mecanismos de diversificação (*shaking* e *lista tabu*), embora o *VNS* tenha apresentado as melhores médias com entradas aleatórias, indicando possíveis limitações nos movimentos de inserção de estações de recarga na fase de intensificação da metaheurística híbrida; (3) o *VND* apresentou piora de desempenho, reforçando as vantagens das metaheurísticas mais sofisticadas.

Entretanto, a alta variabilidade das soluções aleatórias pode comprometer a consistência dos dados, sugerindo a necessidade de ampliar o número de testes em trabalhos futuros para validação estatística mais robusta.

Por último, baseando nas distâncias das tabelas 4 e 5, e considerando que os resultados obtidos se aproximam do ótimo global para o problema do caixeiro viajante elétrico com janelas de tempo (*PCVEJT*), é possível concluir que as distâncias das metaheurísticas para o *PCVE* tiveram uma convergência bem satisfatória, tendo em vista que suas distâncias sobressairam em relação ao trabalho de [Roberti e Wen \(2016\)](#), porém não se deve ignorar as restrições de janela de tempo do problema e a notória diferença em tempo computacional.

As Tabelas 8 e 9 evidenciam que os métodos mais complexos implementados nas metaheurísticas impactaram significativamente o tempo computacional. Este resultado indica a possibilidade de cálculos redundantes nestes algoritmos, reforçando a necessidade de desenvolver e testar novos métodos de critérios de aspiração mais inteligentes, bem como ajustar os critérios de parada. Esse padrão se repete nas Tabelas 12 e 13, onde os algoritmos demandam maior custo computacional em decorrência do aumento no número de iterações necessárias para convergência.

6.2 150 CLIENTES

Tabela 14 – 3P-Heu para instâncias com 150 clientes e 4 estações [Roberti e Wen \(2016\)](#)

Instância	custo	D
n150w120s5.1	753.3	602.6
n150w120s5.2	665.3	532.2
n150w120s5.3	772.1	617.7
n150w120s5.4	738.6	590.9
n150w120s5.5	712.8	570.2
n150w140s5.1	757.6	606.1
n150w140s5.2	773.9	619.1
n150w140s5.3	635.0	508.0
n150w140s5.4	687.0	549.6
n150w140s5.5	674.0	539.2
n150w160s5.1	735.0	588.0
n150w160s5.2	712.9	570.3
n150w160s5.3	639.4	511.5
n150w160s5.4	693.4	554.7
n150w160s5.5	677.9	542.3
Média Geral	707.3	565.8

Tabela 15 – 3P-Heu para instâncias com 150 clientes e 9 estações [Roberti e Wen \(2016\)](#)

Instância	custo	D
n150w120s10.1	746.7	597.4
n150w120s10.2	654.7	523.8
n150w120s10.3	768.3	614.6
n150w120s10.4	722.9	578.3
n150w120s10.5	694.6	555.7
n150w140s10.1	747.3	597.8
n150w140s10.2	768.0	614.4
n150w140s10.3	628.2	502.6
n150w140s10.4	683.0	546.4
n150w140s10.5	673.0	538.4
n150w160s10.1	713.0	570.4
n150w160s10.2	704.1	563.3
n150w160s10.3	630.2	504.2
n150w160s10.4	686.4	549.1
n150w160s10.5	676.1	540.9
Média Geral	699.4	559.5

Solução inicial NN

Tabela 16 – Resultados de distâncias para instâncias de 150 clientes e 9 estações com solução inicial NN

Instância	VND			VNS			VNS/TS			NN
	Melhor D	μ D	(%)	Melhor D	μ D	(%)	Melhor D	μ D	(%)	μ D
n150w120s10.1	552	565.33	7.47	471	476.00	22.09	470	481.33	21.22	611
n150w120s10.2	519	537.67	8.72	456	461.00	21.73	474	482.00	18.17	589
n150w120s10.3	529	536.00	2.01	468	486.33	11.09	479	496.00	9.32	547
n150w120s10.4	517	549.67	5.72	455	474.67	18.58	476	488.00	16.30	583
n150w120s10.5	541	551.67	6.02	480	486.67	17.09	463	476.33	18.85	587
n150w140s10.1	553	567.33	2.02	469	469.67	18.88	485	485.67	16.12	579
n150w140s10.2	552	585.00	0.34	486	494.67	15.73	504	510.33	13.06	587
n150w140s10.3	544	559.00	5.73	458	464.67	21.64	488	488.33	17.65	593
n150w140s10.4	528	533.33	3.90	489	493.67	11.05	499	502.33	9.49	555
n150w140s10.5	524	534.33	9.13	483	487.67	17.06	459	464.00	21.09	588
n150w160s10.1	528	555.67	5.18	464	469.00	19.97	463	470.67	19.68	586
n150w160s10.2	552	572.33	5.09	487	493.00	18.24	484	484.00	19.73	603
n150w160s10.3	571	581.67	9.40	465	485.00	24.45	499	506.33	21.13	642
n150w160s10.4	505	524.33	3.44	461	464.33	14.49	470	472.67	12.95	543
n150w160s10.5	564	579.67	1.75	478	482.33	18.25	472	481.00	18.47	590
Média Geral		555.53	5.06		479.24	18.02		485.93	16.88	585.53

Tabela 17 – Resultados de distâncias para instâncias de 150 clientes e 4 estações com solução inicial NN

Instância	VND			VNS			VNS/TS			NN
	Melhor D	μ D	(%)	Melhor D	μ D	(%)	Melhor D	μ D	(%)	μ D
n150w120s5.1	544	558.33	9.21	467	475.00	22.76	479	481.33	21.73	615
n150w120s5.2	514	537.67	6.82	467	477.33	17.27	463	471.00	18.37	577
n150w120s5.3	567	571.67	0.92	475	481.00	16.64	479	492.33	14.67	577
n150w120s5.4	506	532.00	7.16	457	463.00	19.20	493	495.67	13.50	573
n150w120s5.5	540	557.67	5.00	483	493.00	16.01	472	494.00	15.84	587
n150w140s5.1	530	538.67	9.77	472	478.00	19.93	474	476.67	20.16	597
n150w140s5.2	540	550.67	2.71	483	492.00	13.07	503	506.33	10.54	566
n150w140s5.3	516	581.67	5.88	459	478.67	22.55	473	477.67	22.71	618
n150w140s5.4	548	569.33	-2.58	477	482.33	13.09	473	478.33	13.81	555
n150w140s5.5	615	625.00	-6.29	487	491.67	16.38	482	482.67	17.91	588
n150w160s5.1	536	577.00	9.28	485	499.67	21.44	497	504.67	20.65	636
n150w160s5.2	551	565.33	6.25	465	476.67	20.95	480	485.67	19.46	603
n150w160s5.3	575	604.67	-1.11	477	486.67	18.62	483	485.67	18.78	598
n150w160s5.4	506	543.33	5.51	471	483.00	16.00	475	475.00	17.39	575
n150w160s5.5	547	560.00	4.92	476	478.33	18.79	488	490.33	16.75	589
Média Geral		564.86	4.92		482.42	18.18		486.48	17.49	590.26

Tabela 18 – Resultados de tempos para instâncias de 150 clientes e 4 estações com solução inicial NN

Instância	VND	VNS	VNS/TS	NN
	μ time (s)	μ time (s)	μ time (s)	time (s)
n150w120s5.1	69.55	589.60	5645.53	0.68
n150w120s5.2	46.71	773.22	6787.12	0.70
n150w120s5.3	32.36	596.58	6457.29	1.10
n150w120s5.4	54.30	640.86	5829.86	0.88
n150w120s5.5	39.66	666.89	5149.76	0.71
n150w140s5.1	54.85	523.76	6060.67	1.47
n150w140s5.2	27.65	536.47	5476.89	1.09
n150w140s5.3	50.58	809.76	12340.47	0.77
n150w140s5.4	58.90	505.22	6746.88	0.69
n150w140s5.5	38.58	829.08	6520.23	0.71
n150w160s5.1	73.09	699.04	9299.10	0.88
n150w160s5.2	56.19	482.03	6789.66	0.83
n150w160s5.3	37.43	600.59	6452.07	0.73
n150w160s5.4	32.90	712.87	7519.70	0.79
n150w160s5.5	64.77	479.51	5401.94	0.68
Média	49.17	629.70	6831.81	0.85

Tabela 19 – Resultados de tempos para instâncias de 150 clientes e 9 estações com solução inicial NN

Instância	VND	VNS	VNS/TS	NN
	μ time (s)	μ time (s)	μ time (s)	time (s)
n150w120s10.1	54.30	658.97	5047.94	0.70
n150w120s10.2	32.76	624.00	5401.24	0.65
n150w120s10.3	38.69	749.69	4880.68	1.01
n150w120s10.4	28.23	495.10	6220.79	0.78
n150w120s10.5	49.66	720.90	5605.55	0.93
n150w140s10.1	35.81	659.86	6025.78	0.73
n150w140s10.2	40.43	461.82	6095.67	0.85
n150w140s10.3	51.48	769.64	7588.95	0.87
n150w140s10.4	39.48	513.10	4796.82	0.89
n150w140s10.5	73.32	599.84	10784.68	0.79
n150w160s10.1	31.05	654.89	7462.16	0.73
n150w160s10.2	37.72	550.06	6161.45	0.55
n150w160s10.3	48.22	683.43	7087.89	0.80
n150w160s10.4	46.90	492.05	6254.57	0.73
n150w160s10.5	58.31	571.48	9289.21	0.66
Média	44.42	613.66	6580.23	0.78

Nota: Tempo médio *tabu Search* do VNS/TS: 430.97sNota: Tempo médio *tabu Search* do VNS/TS: 430.97s

Solução inicial aleatória

Tabela 20 – Resultados de distâncias para instâncias de 150 clientes e 9 estações com solução inicial Random

Instância	VND			VNS			VNS/TS			Random μ D
	Melhor D	μ D	(%)	Melhor D	μ D	(%)	Melhor D	μ D	(%)	
n150w120s10.1	589	639.00	99.39	539	551.00	93.48	496	499.00	89.05	54469.25
n150w120s10.2	568	617.25	92.96	530	547.50	88.21	476	478.00	89.28	37868.17
n150w120s10.3	633	651.50	92.46	551	564.50	87.20	517	526.50	99.50	71106.50
n150w120s10.4	579	623.75	99.54	557	569.00	93.44	518	529.00	88.90	71308.25
n150w120s10.5	592	637.00	92.47	508	533.00	94.11	482	486.50	88.73	37795.75
n150w140s10.1	568	580.75	92.97	533	540.50	93.27	483	497.00	88.82	45980.08
n150w140s10.2	630	669.50	99.36	574	586.00	99.44	517	524.00	93.81	87960.92
n150w140s10.3	604	635.00	93.16	560	568.00	93.50	488	497.00	94.37	54630.42
n150w140s10.4	575	601.00	86.43	539	547.50	87.85	475	499.50	94.30	21139.17
n150w140s10.5	617	627.25	92.53	544	549.50	93.19	493	503.50	99.64	87658.42
n150w160s10.1	610	628.00	96.19	545	556.50	93.50	488	508.50	88.74	46101.50
n150w160s10.2	617	634.00	89.16	526	526.50	88.19	506	510.00	88.57	12748.92
n150w160s10.3	604	625.25	92.96	558	565.00	93.41	496	500.50	99.76	104538.67
n150w160s10.4	583	595.50	89.76	523	534.00	93.99	471	478.00	94.64	46106.58
n150w160s10.5	591	642.00	89.19	534	534.50	88.52	482	486.50	89.66	12988.33
Média Geral		627.12	93.24		551.53	92.09		501.57	92.52	

Tabela 21 – Resultados de distâncias para instâncias de 150 clientes e 4 estações com solução inicial Random

Instância	VND			VNS			VNS/TS			Random μ D
	Melhor D	μ D	(%)	Melhor D	μ D	(%)	Melhor D	μ D	(%)	
n150w120s5.1	616	632.75	95.89	555	558.00	99.68	557	572.50	99.64	195915.42
n150w120s5.2	600	635.75	95.93	506	531.50	93.63	525	536.50	87.94	62799.50
n150w120s5.3	592	612.75	92.72	571	571.50	93.69	560	569.50	93.37	87730.17
n150w120s5.4	560	612.50	96.10	557	558.50	87.88	560	563.00	99.66	137921.50
n150w120s5.5	610	645.25	99.67	547	559.00	93.43	549	553.50	94.13	146243.00
n150w140s5.1	550	590.75	96.42	548	548.50	99.80	557	558.00	93.56	212593.92
n150w140s5.2	625	637.75	89.26	564	570.50	99.59	571	573.00	93.75	79503.17
n150w140s5.3	590	632.75	99.46	532	556.00	99.47	562	563.00	93.83	96290.67
n150w140s5.4	573	606.00	96.24	535	550.00	99.73	522	529.00	99.74	187693.92
n150w140s5.5	616	637.50	99.56	560	560.50	93.50	514	522.50	99.83	179403.17
n150w160s5.1	609	632.75	96.15	570	570.00	93.75	568	584.00	93.18	121142.08
n150w160s5.2	569	604.50	99.52	569	575.00	99.72	531	538.00	99.66	187660.58
n150w160s5.3	617	638.50	96.32	540	558.50	99.73	567	570.50	99.63	179524.17
n150w160s5.4	581	616.75	92.89	539	549.00	99.47	514	517.00	99.63	104418.92
n150w160s5.5	551	628.50	99.57	561	566.00	93.39	533	544.50	99.65	146206.17
Média Geral		624.32	96.38		558.83	96.43		552.97	96.48	

Tabela 22 – Resultados de tempos para instâncias de 150 clientes e 9 estações com solução inicial Random

Instância	VND	VNS μ time (s)	VNS/TS
n150w120s10.1	460.81	4925.40	7066.66
n150w120s10.2	482.63	10427.81	15037.65
n150w120s10.3	503.49	9179.46	8255.17
n150w120s10.4	518.60	5764.44	8251.43
n150w120s10.5	503.49	6562.02	11573.65
n150w140s10.1	518.60	7489.25	8483.93
n150w140s10.2	416.07	6259.28	7553.10
n150w140s10.3	419.24	7191.08	13248.50
n150w140s10.4	518.05	7373.18	13825.63
n150w140s10.5	516.13	5503.54	8408.96
n150w160s10.1	507.65	7714.72	6018.65
n150w160s10.2	511.70	12471.14	9500.56
n150w160s10.3	531.58	4697.17	8142.55
n150w160s10.4	503.65	9839.69	8307.81
n150w160s10.5	435.05	12726.89	7487.19
Média	489.78	7875.00	9410.76

Tabela 23 – Resultados de tempos para instâncias de 150 clientes e 4 estações com solução inicial Random

Instância	VND	VNS μ time (s)	VNS/TS
n150w120s5.1	508.46	7858.44	9588.59
n150w120s5.2	535.13	9606.35	7518.95
n150w120s5.3	491.04	9516.41	7969.26
n150w120s5.4	494.95	15249.57	21842.95
n150w120s5.5	520.79	5303.00	12791.12
n150w140s5.1	485.27	7733.74	10977.37
n150w140s5.2	448.44	4106.38	15888.46
n150w140s5.3	538.14	7712.92	10374.84
n150w140s5.4	545.79	8298.85	8283.50
n150w140s5.5	483.58	7605.13	18005.24
n150w160s5.1	382.99	7487.95	7091.30
n150w160s5.2	481.72	8375.69	14079.45
n150w160s5.3	461.99	4082.42	10516.05
n150w160s5.4	550.39	7758.14	10892.76
n150w160s5.5	529.90	8674.90	10051.85
Média	497.24	7957.99	11724.78

Nota: Tempo médio *tabu Search* do VND: 291.09sNota: Tempo médio *tabu Search* do VNS/TS: 303.80s

Análise dos resultados

O cenário de grandes instâncias impactou diretamente nos resultados de 150 clientes em relação a 20, com base nas tabelas 16, 17, 20 e 21, é notório a melhora de resolução da metaheurística *VNS*, comparado com a análise em 6.1, a *VNS* não apenas aumentou sua margem de melhora sobre a *VND*, como se sobressaiu nas tabelas de solução inicial *NN* da *VNS/TS* em torno de 1%, contudo, ficando para trás nos testes de solução aleatória, quase se igualando para 4 estações, mas com uma diferença significativa de 10% em 9 estações.

Dados sobre a melhora de solução para os testes de solução aleatória foram desconsiderados, devido os algoritmos não compartilharem o mesmo ponto de partida.

Na seção anterior 6.1, foi feita a observação sobre o aumento de tempo computacional, que os algoritmos tiveram mais dificuldade de convergência para soluções iniciais aleatórias, o mesmo se repete nas tabelas 22 e 23, porém os algoritmos *VND* e *VNS* tiveram um aumento excessivo maior que 1000%, com o esperado de 35%, mostrando as limitações da *VND* (também utilizada na metaheurística *VNS*), para solucionar problemas com ponto de partida ruins.

Enquanto a *VNS/TS*, que apesar de aumentar seu tempo computacional, o tempo médio das *tabu searches* diminuíram, indicando dois pontos cruciais: (1) facilidade da busca tabu em encontrar movimentos de melhoria em rotas piores; (2) grande aumento de iterações no componente *VNS* da metaheurística híbrida.

Por último, comparando as distâncias em 14 e 15, ambas as metaheurísticas conseguiram superar as distâncias do artigo de Roberti e Wen (2016), com resultados similares para entrada aleatória, e uma margem considerável para entrada *NN*.

6.3 200 CLIENTES

Tabela 24 – 3P-Heu para instâncias com 200 clientes e 4 estações Roberti e Wen (2016)

Instância	custo	D
n200w120s5.1	818.3	654.6
n200w120s5.2	748.6	598.9
n200w120s5.3	900.8	720.6
n200w120s5.4	790.9	632.7
n200w120s5.5	866.0	692.8
n200w140s5.1	836.3	669.0
n200w140s5.2	799.9	639.9
n200w140s5.3	765.0	612.0
n200w140s5.4	821.0	656.8
n200w140s5.5	838.0	670.4
Média Geral	817.7	654.2

Tabela 25 – 3P-Heu para instâncias com 200 clientes e 9 estações Roberti e Wen (2016)

Instância	custo	D
n200w120s10.1	806.1	644.9
n200w120s10.2	738.1	590.5
n200w120s10.3	891.5	713.2
n200w120s10.4	790.9	632.7
n200w120s10.5	857.8	686.2
n200w140s10.1	833.4	666.7
n200w140s10.2	789.8	631.8
n200w140s10.3	764.5	611.6
n200w140s10.4	820.0	656.0
n200w140s10.5	830.9	664.7
Média Geral	810.1	648.1

Solução inicial NN

Tabela 26 – Resultados de distâncias para instâncias de 200 clientes e 9 estações com solução inicial NN

Instância	VND			VNS			VNS/TS			NN
	Melhor D	μ D	(%)	Melhor D	μ D	(%)	Melhor D	μ D	(%)	μ D
n200w120s10.1	696	722.00	-6.02	555	563.00	17.33	560	569.00	16.45	681
n200w120s10.2	624	669.33	2.14	540	551.00	19.44	557	570.67	16.57	684
n200w120s10.3	645	658.33	7.02	556	559.00	21.05	541	545.67	22.93	708
n200w120s10.4	596	620.67	4.81	524	529.67	18.76	545	547.33	16.05	652
n200w120s10.5	632	672.33	6.88	538	545.67	24.42	527	536.33	25.72	722
n200w140s10.1	695	702.67	3.74	533	551.00	24.52	544	552.00	24.38	730
n200w140s10.2	637	650.33	4.50	545	551.33	19.04	558	565.67	16.94	681
n200w140s10.3	675	707.67	-5.46	539	546.67	18.53	525	537.33	19.92	671
n200w140s10.4	605	645.00	-4.71	519	534.00	13.31	508	525.67	14.66	616
n200w140s10.5	639	648.67	2.46	540	552.00	16.99	551	556.00	16.39	665
Média Geral		669.7	1.54		548.33	19.34		550.56	19.00	681

Tabela 27 – Resultados de distâncias para instâncias de 200 clientes e 4 estações com solução inicial NN

Instância	VND			VNS			VNS/TS			NN
	Melhor D	μ D	(%)	Melhor D	μ D	(%)	Melhor D	μ D	(%)	μ D
n200w120s5.1	633	659.00	-0.46	551	563.00	14.18	541	546.00	16.77	656
n200w120s5.2	629	644.67	2.32	535	544.33	17.53	549	555.67	15.81	660
n200w120s5.3	653	670.67	5.54	533	552.00	22.25	534	545.00	23.24	710
n200w120s5.4	639	650.33	0.10	531	544.67	16.33	523	536.67	17.56	651
n200w120s5.5	634	660.33	8.41	542	552.00	23.44	524	537.67	25.43	721
n200w140s5.1	619	648.67	14.98	555	565.00	25.95	559	569.67	25.34	763
n200w140s5.2	671	681.33	2.53	543	547.67	21.65	565	571.67	18.22	699
n200w140s5.3	593	629.00	2.78	532	547.00	15.46	533	541.67	16.28	647
n200w140s5.4	605	618.67	-0.11	514	522.33	15.48	527	539.67	12.68	618
n200w140s5.5	656	667.33	5.88	545	556.00	21.58	551	556.67	21.49	709
Média Geral		653	4.20		549.4	19.38		550.03	19.28	683.4

Tabela 28 – Resultados de tempos para instâncias de 200 clientes e 9 estações com solução inicial NN

Instância	VND	VNS	VNS/TS	NN
	μ time (s)	μ time (s)	μ time (s)	μ time (s)
n200w120s10.1	50.39	2058.81	10360.66	1.15
n200w120s10.2	124.15	2022.78	10161.58	1.43
n200w120s10.3	99.53	1496.44	10046.22	1.06
n200w120s10.4	166.87	1664.35	9896.48	1.71
n200w120s10.5	94.66	1805.14	12587.76	1.28
n200w140s10.1	93.84	1823.59	14623.17	0.75
n200w140s10.2	112.80	1233.37	9647.21	1.43
n200w140s10.3	42.90	1427.30	10780.54	1.34
n200w140s10.4	74.95	2225.89	11433.90	1.17
n200w140s10.5	93.74	1876.76	9147.11	0.60
Média	95.38	1763.44	10868.46	1.19

Nota: Tempo médio *tabu Search* do VNS/TS: 727.08s

Tabela 29 – Resultados de tempos para instâncias de 200 clientes e 4 estações com solução inicial NN

Instância	VND	VNS	VNS/TS	NN
	μ time (s)	μ time (s)	μ time (s)	μ time (s)
n200w120s5.1	97.11	1711.94	11814.37	1.61
n200w120s5.2	126.45	2224.65	12321.56	1.39
n200w120s5.3	83.05	2138.21	12089.51	1.34
n200w120s5.4	70.17	1782.13	12803.93	1.59
n200w120s5.5	88.73	1966.96	8621.98	1.46
n200w140s5.1	184.11	1682.52	12900.61	1.56
n200w140s5.2	96.33	2049.33	11379.33	0.70
n200w140s5.3	98.71	1565.02	12084.20	1.62
n200w140s5.4	53.31	1792.21	9300.78	1.30
n200w140s5.5	121.33	1975.41	10669.79	1.35
Média	101.93	1888.84	11398.61	1.39

Nota: Tempo médio *tabu Search* do VNS/TS: 774.79s

Solução inicial aleatória

Tabela 30 – Resultados de distâncias para instâncias de 200 clientes e 9 estações com solução inicial Random

Instância	VND			VNS			VNS/TS			Random μ D
	Melhor D	μ D	(%)	Melhor D	μ D	(%)	Melhor D	μ D	(%)	
n200w120s10.1	792	829.50	92.88	649	661.50	99.52	598	602.50	90.55	97724.75
n200w120s10.2	755	801.50	95.90	665	676.50	93.95	570	595.00	99.47	97679.67
n200w120s10.3	829	848.25	89.60	650	667.50	89.26	592	599.00	99.44	47724.17
n200w120s10.4	760	806.50	89.58	652	666.50	94.25	567	582.50	99.72	114251.58
n200w120s10.5	771	806.00	92.81	654	660.50	88.32	552	552.50	99.73	97420.92
n200w140s10.1	773	839.75	89.95	682	682.50	99.57	613	616.50	89.97	97851.42
n200w140s10.2	816	826.75	89.51	686	698.00	88.17	618	644.00	99.70	80943.17
n200w140s10.3	788	852.75	92.13	675	675.00	99.36	568	576.00	90.29	72518.58
n200w140s10.4	755	796.25	93.04	659	663.00	88.13	531	531.00	99.50	55543.83
n200w140s10.5	831	887.25	91.91	629	640.00	99.59	602	609.00	89.69	89338.75
Média Geral		829.45	91.73		669.10	94.01		590.80	95.81	

Tabela 31 – Resultados de distâncias para instâncias de 200 clientes e 4 estações com solução inicial Random

Instância	VND			VNS			VNS/TS			Random μ D
	Melhor D	μ D	(%)	Melhor D	μ D	(%)	Melhor D	μ D	(%)	
n200w120s5.1	801	827.75	95.91	646	670.00	88.32	664	695.50	99.66	97509.25
n200w120s5.2	793	831.25	99.64	642	655.00	99.58	676	680.00	99.33	222541.50
n200w120s5.3	775	815.75	99.52	632	645.50	99.59	559	592.50	99.73	197644.92
n200w120s5.4	799	836.50	96.27	660	670.50	99.73	680	680.00	99.36	172446.33
n200w120s5.5	751	797.25	99.51	644	644.00	94.36	647	647.00	99.87	264164.42
n200w140s5.1	817	877.75	96.21	691	702.50	99.66	680	680.00	99.78	247405.75
n200w140s5.2	807	836.50	95.98	694	701.00	99.58	704	712.50	88.06	130826.00
n200w140s5.3	841	862.25	96.14	676	680.50	99.72	658	658.00	99.78	247574.25
n200w140s5.4	776	824.00	95.90	618	622.50	89.28	627	632.50	99.69	105821.83
n200w140s5.5	812	853.75	99.58	659	672.00	99.75	670	672.00	88.74	189186.17
Média Geral		836.28	97.46		666.35	96.96		665.00	97.40	

Tabela 32 – Resultados de tempos para instâncias de 200 clientes e 9 estações com solução inicial Random

Instância	VND	VNS	VNS/TS
	μ time (s)	μ time (s)	μ time (s)
n200w120s10.1	694.48	15520.45	30224.89
n200w120s10.2	646.59	16089.34	22860.61
n200w120s10.3	553.36	15325.56	29520.55
n200w120s10.4	511.16	19397.79	28991.38
n200w120s10.5	526.68	11966.20	32359.67
n200w140s10.1	631.77	20395.04	27426.21
n200w140s10.2	562.12	11379.15	19940.56
n200w140s10.3	537.24	18852.49	28299.39
n200w140s10.4	568.55	13622.46	26743.40
n200w140s10.5	550.81	14423.99	26808.71
Média	578.28	15697.25	27317.54

Nota: Tempo médio *tabu Search* do VNS/TS:
1108.75s

Tabela 33 – Resultados de tempos para instâncias de 200 clientes e 4 estações com solução inicial Random

Instância	VND	VNS	VNS/TS
	μ time (s)	μ time (s)	μ time (s)
n200w120s5.1	737.97	19808.59	31391.56
n200w120s5.2	580.04	23356.76	17792.70
n200w120s5.3	511.93	14764.59	29949.57
n200w120s5.4	563.70	20304.39	33749.72
n200w120s5.5	550.20	16656.27	33294.04
n200w140s5.1	616.03	11270.64	18658.84
n200w140s5.2	628.52	19976.00	26377.94
n200w140s5.3	540.99	20860.99	33777.42
n200w140s5.4	514.80	17308.59	19937.92
n200w140s5.5	566.11	16071.48	44237.85
Média	581.03	18037.83	28916.75

Nota: Tempo médio *tabu Search* do VNS/TS:
1064.49s

Análise dos resultados

Com base nas tabelas 26 e 27, novamente pode se observar a grande vantagem das metaheurísticas, retornando distâncias com melhorias de rota em torno 18% acima da heurística VND, mas reforçando a falta de mecanismo de inserção de estações na etapa de intensificação da VNS/TS. Outro ponto importante, ambos os resultados das tabelas de 4 e 9 estações, as diferenças não são expressivas o suficiente, indicando incapacidade dos algoritmos de usar as estações extras de forma eficiente.

O mesmo padrão observado em 6.2, sobre os resultados de distância da *VNS*, se mantém para 200 clientes, retornando os melhores resultados e novamente tendo uma desvantagem significativa apenas na tabela 30 de solução inicial aleatória com 9 estações de recarga.

As tabelas de tempo 28 e 29, tiveram um comportamento similar a tabela de 150 clientes. Com um aumento em torno de 100% a 150% para os algoritmos *VND* e *VNS* e 50% para a *VNS/TS*, porém essa diferença acontece devido a diminuição de iterações da metaheurística híbrida, tendo em vista que o componente *tabu search* teve um aumento em torno dos 90% de tempo computacional.

Esse aumento brusco de tempo, apesar de adicionar apenas 50 clientes (aumento de 33%), cria implicações futuras para testes maiores. No artigo de Schneider, Stenger e Goeke (2014), a base teórica desse trabalho, foram feitos testes de 300 a 500 clientes, com tempos computacionais similares aos obtidos nesse trabalho.

Em relação a diferença de tempo entre as tabelas 32 e 33 com solução inicial aleatória, novamente os algoritmos *VND* e *VNS* tiveram um aumento um excessivo em torno dos 800%, o mesmo ocorreu, porém em escala menor, na metaheurística híbrida *VNS/TS* com um aumento em torno dos 150%, que diferente dos testes de 150 clientes, dessa vez a *tabu search* aumentou seu tempo médio.

7 CONCLUSÕES

Este trabalho apresentou uma abordagem híbrida para o Problema do Caixeiro Viajante com Veículos Elétricos (PCV-E), combinando as metaheurísticas *Variable Neighborhood Search* (VNS) e *Tabu Search* (TS). A solução proposta, deve teoricamente demonstrar ser eficiente na resolução das complexidades adicionais impostas pelas restrições de autonomia de bateria e necessidades de recarga, características fundamentais dos veículos elétricos.

Os resultados computacionais das instâncias mostraram que a nova formulação proposta é capaz de resolver instâncias de 20 clientes, validando restrições de capacidade de bateria em rotas com distâncias satisfatórias. A metaheurística híbrida foi capaz de converter esses bons resultados tanto para instâncias de larga escala, de 150 a 200 clientes.

Os dados reforçam o potencial das metaheurísticas em enfrentar os desafios logísticos para veículos elétricos, as principais contribuições sendo: (1) Soluções de distâncias satisfatórias, os algoritmos *VNS* e *VNS/TS* foram capazes de gerar resultados superiores em relação ao trabalho de [Roberti e Wen \(2016\)](#), se considerar apenas as restrições de capacidade de bateria e distância final; (2) A combinação de métodos construtivos e heurísticas se mostrou favorável, a solução inicial *S* impacta diretamente na convergência dos algoritmos, tanto na qualidade das soluções como no tempo computacional, por isso, os métodos construtivos e seu baixo custo computacional são de grande importância para a criação de bons métodos para o problema. (3) A complexidade do Problema do Caixeiro Viajante (NP-difícil), impossibilita a capacidade de se obter resultados exatos para grandes instâncias em tempo justificável, por isso, a necessidade em buscar métodos alternativos que convergem em soluções aceitáveis, reforçam as vantagens das metaheurísticas como método de resolução. (4) A tentativa de juntar diferentes metaheurísticas, mostrou em alguns casos grandes melhorias, porém com um tempo computacional elevado, o que poderia não justificar a mesclagem. No trabalho de [Schneider, Stenger e Goeke \(2014\)](#) são utilizadas outras estratégias como *simulated annealing SA* e um *Tabu Search TS* mais elaborado, indicando as limitações da *VNS/TS* desenvolvida neste projeto.

7.1 TRABALHOS FUTUROS

Este trabalho visa compreender o funcionamento de metaheurísticas, algoritmos construtivos, suas vantagens e desvantagens de uso, e combinar diferentes estratégias buscando melhorias justificáveis para a junção, como mostrado no capítulo 7, há espaço para muitas melhorias em todos os algoritmos testados: *Nearest Neighbor NN*; *Variable Neighborhood Descent VND*; *Variable Neighborhood Search VNS*; *Tabu Search TS*; e por último o principal abordado neste trabalho *hybrid VNSTS*.

Apesar do foco deste trabalho não focar explicitamente em desempenho e eficiência máxima, alguns quesitos poderiam ser melhoradas:

Como direções futuras, sugere-se:

- Aumentar número de testes para garantir resultados mais complexos e testes de eficácia mais completas.
- Ajustar parâmetros das metaheurísticas para máxima eficiência.
- Criação de um critério de aspiração mais complexos, utilizando de estratégias como *Simulated Annealing SA*, usando fatores de temperatura e resfriamento, esta estratégia pode ser vista sendo utilizada no artigo de [Schneider, Stenger e Goeke \(2014\)](#).
- Testar diferentes estratégias de perturbação para o VNS, diferente de como foi implementado neste trabalho, com uso de graus de intensidade e escolhas aleatórias dos movimentos *2-opt*, *swap*, *descolate forward/backward* e *station insertion/removal*, testar a criação de vizinhanças utilizando movimentos unicos e com a troca periodica dos tipos de movimento.
- Testar diferentes parâmetros e movimentos da *Tabu Search*, testar vizinhanças de movimentos variados e não apenas *2-opt*, o que traria novas formas de se implementar movimentos tabus e configurações de lista tabu, e por ultimo comparar diferentes resultados de parâmetros variados de *tabu tenure* e diferentes formas de lidar com movimentos tabus.
- Estudar e implementar novas estratégias de inserção mais eficientes de estações de recarga com métodos de programação dinâmica.
- Evoluir o problema para lidar com janelas de tempo como demonstrado pelo artigo de [Roberti e Wen \(2016\)](#) sobre o *Electric Traveling Salesman Problem with Time Windows ETSPW*.

REFERÊNCIAS

CESELLI, Alberto; RIGHINI, Giovanni. **The Electric Traveling Salesman Problem: properties and models**. [S.l.: s.n.], 2020. Research Gate. Disponível em: <https://www.researchgate.net/publication/345976231_The_Electric_Traveling_Salesman_Problem_properties_and_models>. Citado na p. 17.

CHRISTOFIDES, Nicos. Worst-case analysis of a new heuristic for the travelling salesman problem. **Graduate School of Industrial Administration, Carnegie-Mellon University**, 1976. Technical Report 388. Disponível em: <<https://apps.dtic.mil/sti/pdfs/ADA025602.pdf>>. Citado nas pp. 22, 23, 29.

DAVIS, Adam. **Traveling Salesman Problem With the 2-opt Algorithm**. [S.l.: s.n.], 2022. Disponível em: <<https://slowandsteadybrain.medium.com/traveling-salesman-problem-ce78187cf1f3>>. Citado na p. 20.

GENDREAU, Michel; POTVIN, Jean-Yves. **Handbook of Metaheuristics**. [S.l.]: Springer, 2010. Citado na p. 24.

HEIMANN, Sophia; HOANG, Hung P.; HOUGARDY, Stefan. **The k-Opt algorithm for the Traveling Salesman Problem has exponential running time for $k \geq 5$** . [S.l.: s.n.], 2024. Disponível em: <<https://arxiv.org/abs/2402.07061>>. Citado na p. 21.

HIERHOLZER, Carl. Über die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren. **Mathematische Annalen**, Springer, v. 6, n. 1, p. 30–32, 1873. Citado na p. 24.

HUANG, Chung-Yang; LAI, Chao Yue; CHENG, Kwang-Ting. **Fundamentos de algoritmos**. [S.l.: s.n.], 2009. Disponível em: <<https://www-sciencedirect-com.translate.goog/science/article/abs/pii/B9780123743640500114>>. Citado na p. 18.

IEA. **Global CO2 emissions from transport by sub-sector**. [S.l.], 2023. Disponível em: <<https://www.iea.org/data-and-statistics/charts/global-co2-emissions-from-transport-by-sub-sector-in-the-net-zero-scenario-2000-2030-2>>. Citado na p. 10.

INTERNATIONAL ENERGY AGENCY. **Electric Vehicles**. [S.l.], 2025. Licence: CC BY 4.0. Disponível em:

<<https://www.iea.org/energy-system/transport/electric-vehicles>>. Citado na p. 9.

KHULLER, Samir; MALEKIAN, Azarakhsh; MESTRE, Julián. To Fill or not to Fill: The Gas Station Problem. *In: PROCEEDINGS of the 18th Annual European Symposium on Algorithms*. [S.l.: s.n.], 2011. P. 415–426. DOI: 10.1007/978-3-540-75520-3_48. Citado na p. 34.

LIAO, C.-S.; LU, S.-H.; SHEN, Z.-J. M. The electric vehicle touring problem. **Transportation Research Part B: Methodological**, v. 86, p. 163–180, 2016. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0191261516000278>>. Citado nas pp. 34–36.

MIYAZAWA, F. K. [S.l.], 2021. Disponível em: <<https://www.ic.unicamp.br/~fkm/lectures/slides-exatos.pdf>>. Citado na p. 10.

NGUYEN, Quang Nhat. **Travelling Salesman Problem and Bellman-Held-Karp Algorithm**. [S.l.: s.n.], 2020. Disponível em: <<https://www.math.nagoya-u.ac.jp/~richard/teaching/s2020/Quang1.pdf>>. Citado na p. 16.

PRIM, Robert C. Shortest connection networks and some generalizations. **Bell System Technical Journal**, Nokia Bell Labs, v. 36, n. 6, p. 1389–1401, 1957. Citado na p. 24.

RANGEL-MONDRAGON, Jaime. [S.l.: s.n.], 2012. Disponível em: <<https://demonstrations.wolfram.com/TheTravelingSalesmanProblem3NearestNeighborHeuristic/>>. Citado na p. 22.

ROBERTI, R.; WEN, M. The Electric Traveling Salesman Problem with Time Windows. **Transportation Research Part B: Methodological**, v. 89, p. 178–201, 2016. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S1366554516000181>>. Citado nas pp. 6, 7, 12–14, 32, 35, 36, 39, 41, 42, 46, 49, 53, 54.

SCHNEIDER, M.; STENGER, A.; GOEKE, D. The Electric Vehicle-Routing Problem with Time Windows and Recharging Stations. **Transportation Science**, v. 48, n. 4, p. 500–520, 2014. Disponível em: <<https://www.jstor.org/stable/43666939>>. Citado nas pp. 11, 12, 29, 30, 35, 37, 52–54.

SILVA, Rodrigo; URRUTIA, Sebastián. A General VNS Heuristic for the Traveling Salesman Problem with Time Windows. **Electronic Notes in Discrete Mathematics**, v. 36, p. 203–211, 2010. DOI: 10.1016/j.endm.2010.05.026. Disponível em:

<<https://www.sciencedirect.com/science/article/pii/S1572528610000289>>.

Citado na p. 32.

SILVEIRA, Porto da. [S.l.], 2000. Disponível em:

<<http://www.mat.ufrgs.br/~portosil/caixeiro.html>>. Citado na p. 15.

SOVIYAN, Munawar. Optimizing Urban Logistics Through Electric Vehicle Integration.

Sinergi International Journal of Logistics, v. 2, n. 3, p. 160–173, 2024. Disponível em:

<<https://journal.sinergi.or.id/>>. Citado na p. 10.

ZÄPFEL, Günther; BRAUNE, Rainer; BÖGL, Michael. **Metaheuristic Search Concepts**.

[S.l.]: Springer, 2009. Citado nas pp. 19, 24.

APÊNDICE A – ARTIGO

*Uma Metaheurística Híbrida VNS/TS para o Problema do
Caixeiro Viajante com Veículos Elétricos*

Uma Metaheurística Híbrida VNS/TS para o Problema do Caixeiro Viajante com Veículos Elétricos

Leonardo Seishi Yamazaki¹, Rafael de Santiago¹

¹Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Campus Trindade – 88040-900 – Florianópolis – SC – Brasil

leonardo.yamazaki@grad.ufsc.br, rafael.santiago@ufsc.br

Abstract. *The growing adoption of electric vehicles (EVs) in logistics introduces new computational challenges in route optimization due to battery autonomy and recharging constraints. This work proposes and evaluates a hybrid metaheuristic combining Variable Neighborhood Search (VNS) and Tabu Search (TS) for the Electric Vehicle Traveling Salesman Problem (E-TSP). The hybrid algorithm, along with Nearest-Neighbor construction, Variable Neighborhood Descent (VND) and standalone VNS baselines, was evaluated on 100 public benchmark instances (20, 150 and 200 customers) proposed by Roberti and Wen. Results show that, starting from a Nearest-Neighbor solution, the hybrid VNS/TS achieved the best route distances for small instances (up to 21.67% improvement over the initial solution), while VNS alone was more effective and consistent for larger instances (150 and 200 customers), at a substantially lower computational cost. All feasible solutions strictly satisfy the battery constraints of the problem.*

Keywords: *Metaheuristics, Electric Vehicle Routing, Traveling Salesman Problem, VNS, Tabu Search.*

Resumo. *A crescente adoção de veículos elétricos (VEs) no setor logístico introduz novos desafios computacionais na otimização de rotas devido a restrições de autonomia de bateria e recarga. Este trabalho propõe e avalia uma metaheurística híbrida combinando Variable Neighborhood Search (VNS) e Tabu Search (TS) para o Problema do Caixeiro Viajante com Veículos Elétricos (PCV-E). O algoritmo híbrido, junto de uma heurística construtiva Vizinho Mais Próximo, uma busca local Variable Neighborhood Descent (VND) e uma metaheurística VNS isolada, foi avaliado em 100 instâncias públicas de benchmark (20, 150 e 200 clientes) propostas por Roberti e Wen. Os resultados mostram que, partindo de uma solução inicial Vizinho Mais Próximo, o híbrido VNS/TS obteve as melhores distâncias para instâncias pequenas (melhoria de até 21,67% sobre a solução inicial), enquanto o VNS isolado se mostrou mais eficaz e consistente para instâncias maiores (150 e 200 clientes), com custo computacional substancialmente menor. Todas as soluções factíveis respeitam rigorosamente as restrições de bateria do problema.*

Palavras-chave: *Metaheurísticas, Roteamento de Veículos Elétricos, Caixeiro Viajante, VNS, Busca Tabu.*

1. Introdução

O avanço da tecnologia de veículos elétricos (VEs) vem transformando o setor de transportes e logística em todo o mundo. Dados da International Energy Agency

[International Energy Agency 2025] indicam que as vendas globais de VEs ultrapassaram 17 milhões de unidades em 2024, representando mais de 20% do mercado automotivo global, com o primeiro trimestre de 2025 apresentando crescimento interanual de 35%. No contexto logístico, estima-se que até 2030 cerca de 30% dos veículos comerciais leves serão elétricos, motivado tanto por pressões ambientais quanto por vantagens econômicas – os VEs já oferecem redução de até 60% nos custos de energia e manutenção por quilômetro rodado em relação a veículos convencionais.

Essa transição traz consigo desafios computacionais inéditos para a otimização de rotas. Conforme discutido por Soviyan [Soviyan 2024], a otimização de rotas para VEs é significativamente mais complexa do que para veículos convencionais devido às restrições de capacidade da bateria, à distribuição desigual e à disponibilidade limitada de estações de recarga, exigindo abordagens especializadas como metaheurísticas híbridas.

Considerando que o setor de transportes rodoviário corresponde a 74% das emissões de CO₂ do setor de energia [IEA 2023], mesmo pequenas otimizações de rota podem gerar impactos ambientais relevantes: uma redução de apenas 1% na distância percorrida por veículos de carga em operações urbanas representaria uma economia anual estimada em 1,2 milhão de toneladas de CO₂ em escala global.

O Problema do Caixeiro Viajante para Veículos Elétricos (PCV-E) estende o clássico Problema do Caixeiro Viajante (PCV) – já NP-difícil por si só – com restrições adicionais de autonomia de bateria e necessidade de recarga em estações específicas. Algoritmos exatos, como o Bellman-Held-Karp, possuem complexidade $O(n^2 \cdot 2^n)$, tornando-se inviáveis para instâncias com mais de poucas dezenas de clientes [Miyazawa 2021]. Diante disso, heurísticas e metaheurísticas bem projetadas tornam-se a abordagem preferencial para problemas de grande escala.

Este trabalho propõe uma metaheurística híbrida para o PCV-E, combinando as vantagens do *Variable Neighborhood Search* (VNS) e da *Tabu Search* (TS), inspirada na estrutura de sucesso demonstrada por Schneider, Stenger e Goeke [Schneider et al. 2014] para o problema correlato E-VRPTW, adaptada para um cenário simplificado de um único veículo e sem janelas de tempo. As principais contribuições deste trabalho são: (1) uma revisão sistemática de soluções da literatura para o PCV-E; (2) o desenvolvimento e implementação de uma heurística híbrida de roteamento que considera restrições de recarga; e (3) uma análise comparativa de desempenho contra métodos construtivos e de busca local clássicos, utilizando benchmarks públicos amplamente reconhecidos na literatura.

2. Definição do Problema

O PCV-E é modelado como um grafo completo dirigido $G = (V, E)$, com $V = C \cup R \cup \{o\}$, onde $C = \{c_1, \dots, c_n\}$ é o conjunto de clientes, $R = \{r_1, \dots, r_m\}$ é o conjunto de estações de recarga e o é o depósito (origem/destino), que, diferentemente de outras formulações da literatura [Roberti and Wen 2016], não atua como estação de recarga neste trabalho. Cada arco $(u, v) \in E$ possui um custo de deslocamento $w(u, v) \geq 0$ e um consumo de energia $e(u, v) \geq 0$.

Deseja-se encontrar uma rota $S = \langle v_1, \dots, v_k \rangle$, com $v_1 = v_k = o$, que visite todos os clientes exatamente uma vez, inserindo estações de recarga conforme necessidade

energética. A rota é, portanto, hamiltoniana em relação a C , mas não necessariamente em relação a V , já que estações podem ser repetidas [Schneider et al. 2014].

Seja $B > 0$ a capacidade máxima da bateria, com $b_0 = B$ (bateria cheia na origem). O nível de energia b_i ao chegar em v_i é atualizado por:

$$b_{i+1} = \begin{cases} B - e(v_i, v_{i+1}) & \text{se } v_i \in R, \\ b_i - e(v_i, v_{i+1}) & \text{caso contrário.} \end{cases} \quad (1)$$

O objetivo é minimizar a distância total da rota acrescida de um custo fixo α por recarga:

$$\min \sum_{i=1}^{k-1} w(v_i, v_{i+1}) + \alpha \cdot |\{v_i \in S : v_i \in R\}| \quad (2)$$

sujeita a: (1) todos os clientes devem ser visitados exatamente uma vez; (2) toda visita a uma estação resulta em recarga completa ($b_i = B$); (3) a energia b_i nunca pode ser negativa; e (4) estações de recarga podem ou não ser visitadas, conforme a necessidade.

As instâncias utilizadas seguem o formato proposto por Roberti e Wen [Roberti and Wen 2016] para o Electric Traveling Salesman Problem with Time Windows (E-TSPTW), contendo número de clientes, número de estações, capacidade de bateria, taxas de consumo e recarga, coordenadas dos nós e a matriz de distâncias/tempos. Para este trabalho, as restrições de janela de tempo dessas instâncias foram desconsideradas, uma vez que o foco está exclusivamente nas restrições de autonomia e recarga.

3. Trabalhos Relacionados

Três trabalhos da literatura fundamentam a abordagem proposta, cada um endereçando uma variante distinta de roteamento para veículos elétricos.

Schneider, Stenger e Goeke [Schneider et al. 2014] propõem uma heurística híbrida VNS/TS para o *Electric Vehicle-Routing Problem with Time Windows and Recharging Stations* (E-VRPTW), um problema de múltiplos veículos com janelas de tempo. No método, o VNS é responsável pela diversificação através de estruturas de vizinhança variáveis, enquanto a Tabu Search substitui a busca local tradicional na etapa de intensificação, amostrando aleatoriamente um subconjunto da vizinhança e utilizando uma lista tabu para evitar ciclos e escapar de ótimos locais. Este trabalho permanece como referência de estado da arte para o E-VRPTW e serviu de base direta para a metaheurística híbrida aqui desenvolvida, simplificada para um único veículo e sem janelas de tempo.

Roberti e Wen [Roberti and Wen 2016] propõem um algoritmo heurístico de três fases para o *Electric Traveling Salesman Problem with Time Windows* (E-TSPTW), inspirado no trabalho de Silva e Urrutia [Silva and Urrutia 2010] para o TSPTW clássico. As fases consistem em: (1) alcançar viabilidade de janela de tempo via *Variable Neighborhood Descent* (VND); (2) busca local para redução de distância, também via VND, explorando vizinhanças de realocação e troca; e (3) viabilização da capacidade de bateria por meio de um algoritmo de programação dinâmica que insere paradas ótimas de recarga.

As 100 instâncias de benchmark (20, 150 e 200 clientes) utilizadas neste trabalho são as mesmas propostas por esse artigo.

Liao, Lu e Shen [Liao et al. 2016] propõem algoritmos de aproximação de fator constante para o *Electric Vehicle Touring Problem*, mostrando que o problema de recarga parcial em estação é equivalente ao *gas station problem* [Khuller et al. 2011], com fatores de aproximação de $9/4$ e $9/2$ para custos uniformes e não uniformes, respectivamente, aproximando-se do fator do algoritmo de Christofides [Christofides 1976] quando a bateria possui capacidade suficientemente alta.

A Tabela 1 resume as diferenças entre os três trabalhos. Optou-se por basear a solução principal na metaheurística híbrida VNS/TS de Schneider, Stenger e Goeke [Schneider et al. 2014] devido à sua adaptabilidade e ao balanceamento comprovado entre diversificação e intensificação, incorporando de Roberti e Wen [Roberti and Wen 2016] a noção de fases para viabilidade de bateria e as instâncias de teste, e considerando de Liao, Lu e Shen [Liao et al. 2016] os limites teóricos de desempenho, por ser o trabalho mais próximo em termos de escopo (único veículo, sem janelas de tempo).

Tabela 1. Comparação entre os trabalhos relacionados

Aspecto	Schneider et al. [Schneider et al. 2014]	Roberti e Wen [Roberti and Wen 2016]	Liao et al. [Liao et al. 2016]
Problema	E-VRPTW	E-TSPTW	EV Touring Problem
N.º de veículos	Múltiplos	Único	Único
Restrições	TW, Bateria, Recarga	TW, Bateria	Bateria
Técnica	VNS/TS híbrido	VND + Prog. Dinâmica	Algoritmos de Aproximação
Instâncias	Até 500 clientes	Até 200 clientes	Teórico/NP-difícil
Contribuição	Metaheurística híbrida	Algoritmo em 3 fases	Fator de aproximação

4. Metodologia

A solução proposta neste trabalho é uma metaheurística híbrida de duas fases que combina VNS e TS, adaptada da estrutura de Schneider, Stenger e Goeke [Schneider et al. 2014] para o cenário simplificado de um único veículo sem janelas de tempo.

4.1. Algoritmo Híbrido VNS/TS

O algoritmo parte de uma solução inicial S_0 – gerada pela heurística construtiva do Vizinho Mais Próximo [Rangel-Mondragon 2012] ou por geração aleatória – e itera por um número máximo de iterações, alternando entre uma fase de diversificação (VNS) e uma fase de intensificação (TS), conforme o Algoritmo 1.

A fase de perturbação do VNS aplica movimentos aleatórios de *deslocate forward*, *deslocate backward*, *swap* e inserção de estação (*station insertion*), com intensidade proporcional a k . Em seguida, uma busca local remove estações de recarga desnecessárias da rota. Por fim, o componente Tabu Search substitui a descida local (VND) convencional utilizada por Schneider et al.: a vizinhança é construída por amostragem aleatória de movimentos 2-opt – em vez de enumerar todas as $O(n^2)$ possibilidades –, e uma lista tabu impede o retorno a movimentos recentes, permitindo aceitar soluções de qualidade inferior quando nenhuma melhoria está disponível. Nesta implementação, diferentemente da formulação teórica clássica da Tabu Search, não foi utilizado um critério de aspiração: o componente TS aceita apenas movimentos não proibidos pela lista tabu.

Algorithm 1 Metaheurística híbrida VNS/TS

Require: Grafo G , bateria B , consumo $e(u, v)$, solução inicial S_0

Ensure: Ciclo S^* factível de custo mínimo

```
1:  $S^* \leftarrow S_0, TabuList \leftarrow \emptyset, k_{max} \leftarrow 6$ 
2: for  $iter \leftarrow 1$  to  $IterMax$  do
3:    $k \leftarrow 1$ 
4:    $S' \leftarrow Perturbação\_VNS(S_{current}, k)$  ▷ 2-opt / swap / insertion
5:    $S'' \leftarrow TabuSearch(S', TabuList)$ 
6:   if  $grade(S'') < grade(S^*)$  then
7:      $S^* \leftarrow S''; k \leftarrow 1$ 
8:   else
9:      $k \leftarrow k + 1$ 
10:  end if
11: end for
12: return  $S^*$ 
```

4.2. Testes e Execução

Foram utilizadas as 100 instâncias públicas propostas por Roberti e Wen [Roberti and Wen 2016]: 50 instâncias de 20 clientes, 30 de 150 clientes e 20 de 200 clientes, divididas igualmente entre configurações de 4 e 9 estações de recarga. Cada algoritmo foi executado 3 vezes para cada instância, implementado em Python 3.10¹, em um computador Intel(R) Xeon(R) CPU @ 3.60GHz com 8 threads e 32GB de RAM. Foram utilizadas duas soluções iniciais – Vizinho Mais Próximo (NN) e geração aleatória – de modo a avaliar tanto a capacidade de melhoria dos algoritmos quanto uma comparação justa a partir do mesmo ponto de partida.

Os algoritmos comparados foram: a heurística construtiva Nearest Neighbor (NN); a busca local determinística Variable Neighborhood Descent (VND); a metaheurística Variable Neighborhood Search (VNS); e a metaheurística híbrida proposta (VNS/TS). As métricas de avaliação foram: (1) tempo de execução em segundos; (2) distância final da rota, garantindo zero violações de bateria; e (3) percentual de melhoria em relação à distância da solução inicial.

5. Resultados e Discussões

Todas as rotas geradas pelos algoritmos VND, VNS e VNS/TS garantem viabilidade quanto à capacidade de bateria. A Tabela 2 resume o percentual médio de melhoria de distância em relação à solução inicial Vizinho Mais Próximo, para os três portes de instância avaliados.

Para instâncias de 20 clientes, o algoritmo híbrido VNS/TS obteve o melhor desempenho, com uma margem de melhoria de 4 a 7 pontos percentuais acima dos demais algoritmos, demonstrando a eficácia da Tabu Search em escapar de ótimos locais nos quais o VND e o VNS convencionais tendem a ficar presos devido à sua dependência de sequências de movimentos aleatórios.

¹Código-fonte disponível em: <https://github.com/LeonardoSeishi/TCC>

Tabela 2. Melhoria média (%) sobre a distância inicial NN, por porte de instância

N.º de clientes	VND	VNS	VNS/TS
20	14,5%	14,9%	21,1%
150	5,0%	18,1%	17,2%
200	2,9%	19,4%	19,1%

Já para instâncias maiores (150 e 200 clientes), o cenário se inverte: o VNS passa a superar consistentemente o VNS/TS, embora por margem pequena (cerca de 1 ponto percentual), o que sugere limitações no mecanismo de inserção de estações de recarga durante a fase de intensificação da metaheurística híbrida em instâncias de maior escala. Em todos os portes, ambas as metaheurísticas superam com folga a busca local determinística (VND), evidenciando a importância dos mecanismos de diversificação para escapar de ótimos locais.

Quando comparados a partir de uma solução inicial completamente aleatória, tanto o VNS quanto o VNS/TS apresentam melhorias percentuais muito superiores (acima de 60% para 20 clientes, e próximas de 90-97% para 150 e 200 clientes), uma vez que partem de soluções de qualidade muito inferior; nesse cenário, contudo, o VNS/TS demonstra maior eficácia em escapar da solução inicial ruim, enquanto o VND apresenta piora relativa de desempenho, reforçando as vantagens das metaheurísticas mais sofisticadas diante de pontos de partida desfavoráveis.

Em relação ao tempo computacional, o custo do VNS/TS é consideravelmente mais alto que o do VND e do VNS isolado, sobretudo pelo elevado número de iterações da Tabu Search: para instâncias de 20 clientes, o tempo médio do VNS/TS foi de aproximadamente 30s, contra 6s do VNS e menos de 1s do VND; para 200 clientes, esse tempo ultrapassa 10.000s (aproximadamente 3 horas), contra cerca de 1.800s do VNS. Esse crescimento expressivo do custo computacional, mesmo com ganhos de qualidade marginais em instâncias grandes, indica que a combinação das duas metaheurísticas nem sempre se justifica computacionalmente, sobretudo quando comparada ao desempenho já satisfatório do VNS isolado.

Por fim, ao comparar as distâncias finais obtidas com os resultados do algoritmo de três fases de Roberti e Wen [Roberti and Wen 2016] – ainda que este resolva um problema com janelas de tempo adicionais –, tanto o VNS quanto o VNS/TS obtiveram distâncias finais menores, sugerindo uma convergência satisfatória das metaheurísticas desenvolvidas para o PCV-E, embora a diferença em tempo computacional e em restrições do problema não deva ser ignorada nessa comparação.

6. Conclusão

Este trabalho apresentou uma abordagem híbrida para o Problema do Caixeiro Viajante com Veículos Elétricos (PCV-E), combinando as metaheurísticas Variable Neighborhood Search (VNS) e Tabu Search (TS). Os experimentos em 100 instâncias públicas de 20, 150 e 200 clientes mostraram que a metaheurística híbrida é particularmente eficaz para instâncias pequenas, enquanto o VNS isolado se mostra mais consistente e computacionalmente mais barato para instâncias de maior escala.

As principais contribuições deste trabalho são: (1) resultados de distância satis-

fatórios, com o VNS e o VNS/TS superando o algoritmo de referência de Roberti e Wen [Roberti and Wen 2016] quanto à distância final, considerando apenas restrições de bateria; (2) evidência de que a qualidade da solução inicial impacta diretamente a convergência dos algoritmos, tanto na qualidade da solução quanto no tempo computacional, reforçando a importância de métodos construtivos de baixo custo; (3) confirmação de que, dada a natureza NP-difícil do problema, metaheurísticas permanecem a abordagem preferencial para instâncias de grande escala; e (4) evidência de que a combinação de metaheurísticas nem sempre compensa o custo computacional adicional, especialmente em instâncias grandes, sugerindo espaço para versões mais elaboradas de critérios de aspiração e de parada, como as empregadas por Schneider, Stenger e Goeke [Schneider et al. 2014].

Como trabalhos futuros, sugere-se: (1) ampliar o número de execuções para maior robustez estatística; (2) incorporar critérios de aspiração mais sofisticados, como estratégias de Simulated Annealing; (3) testar novas estruturas de vizinhança e parâmetros para o VNS e a Tabu Search; (4) desenvolver estratégias de inserção de estações de recarga baseadas em programação dinâmica; e (5) estender o modelo para considerar janelas de tempo, como no E-TSPTW de Roberti e Wen [Roberti and Wen 2016].

Referências

- [Christofides 1976] Christofides, N. (1976). Worst-case analysis of a new heuristic for the travelling salesman problem. *Graduate School of Industrial Administration, Carnegie-Mellon University*. Technical Report 388.
- [IEA 2023] IEA (2023). Global co2 emissions from transport by sub-sector. Technical report, IEA.
- [International Energy Agency 2025] International Energy Agency (2025). Electric vehicles. Technical report, IEA. Licence: CC BY 4.0.
- [Khuller et al. 2011] Khuller, S., Malekian, A., and Mestre, J. (2011). To fill or not to fill: The gas station problem. In *Proceedings of the 18th Annual European Symposium on Algorithms*, pages 415–426.
- [Liao et al. 2016] Liao, C.-S., Lu, S.-H., and Shen, Z.-J. M. (2016). The electric vehicle touring problem. *Transportation Research Part B: Methodological*, 86:163–180.
- [Miyazawa 2021] Miyazawa, F. K. (2021). Complexidade de algoritmos: Métodos exatos. Technical report, Unicamp.
- [Rangel-Mondragon 2012] Rangel-Mondragon, J. (2012).
- [Roberti and Wen 2016] Roberti, R. and Wen, M. (2016). The electric traveling salesman problem with time windows. *Transportation Research Part B: Methodological*, 89:178–201.
- [Schneider et al. 2014] Schneider, M., Stenger, A., and Goeke, D. (2014). The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, 48(4):500–520.
- [Silva and Urrutia 2010] Silva, R. and Urrutia, S. (2010). A general VNS heuristic for the traveling salesman problem with time windows. *Electronic Notes in Discrete Mathematics*, 36:203–211.

[Soviyan 2024] Soviyan, M. (2024). Optimizing urban logistics through electric vehicle integration. *Sinergi International Journal of Logistics*, 2(3):160–173.

APÊNDICE B – REPOSITÓRIO

O código-fonte associado a este trabalho encontra-se organizado no repositório abaixo. Para reprodução dos experimentos principais, utilizar as instruções presentes no arquivo README.md.

- <https://github.com/LeonardoSeishi/TCC>