UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO DE JOINVILLE
CURSO DE ENGENHARIA DE TRANSPORTES E LOGÍSTICA

JAQUELINE CRISTINA DA ROSA

MODELO DE OTIMIZAÇÃO PARA O PLANEJAMENTO LOGÍSTICO DO PROCESSO DE PINTURA DE SINALIZAÇÃO HORIZONTAL EM VIAS URBANAS NO DEPARTAMENTO DE TRÂNSITO DO MUNICÍPIO DE JOINVILLE

Joinville
2025

JAQUELINE CRISTINA DA ROSA

MODELO DE OTIMIZAÇÃO PARA O PLANEJAMENTO LOGÍSTICO DO PROCESSO DE PINTURA DE SINALIZAÇÃO HORIZONTAL EM VIAS URBANAS NO DEPARTAMENTO DE TRÂNSITO DO MUNICÍPIO DE JOINVILLE

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do título de Bacharel em Engenharia de Transportes e Logística, no curso Engenharia de Transportes e Logística da Universidade Federal de Santa Catarina, Centro Tecnológico de Joinville.

Orientadora: Prof. Dra. Silvia Lopes de Sena Taglialenha

Joinville
2025

JAQUELINE CRISTINA DA ROSA

MODELO DE OTIMIZAÇÃO PARA O PLANEJAMENTO LOGÍSTICO DO PROCESSO DE PINTURA DE SINALIZAÇÃO HORIZONTAL EM VIAS URBANAS NO DEPARTAMENTO DE TRÂNSITO DO MUNICÍPIO DE JOINVILLE

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do título de bacharel em Engenharia de Transportes e Logística, na Universidade Federal de Santa Catarina, Centro Tecnológico de Joinville.

Joinville (SC), 18 de novembro de 2025.

Banca Examinadora:

Dra. Silvia Lopes de Sena Taglialenha
Orientador(a)
Presidente

Dr. Rômulo Alberto Castillo Cardenas
Membro(a)
Universidade Federal de Santa Catarina

Eng. Samuel Luiz Bernardes Gomes
Membro(a)
Detrans-Joinville

Para aqueles que me ensinaram a voar, dedico o meu primeiro voo solo.

AGRADECIMENTOS

Dedico esta conquista, em primeiro lugar, à minha família. Aos meus pais, Elaine e Claudinei, por todo o amor, sacrifício e por serem a base de tudo o que sou. Suas palavras de incentivo foram meu combustível. À minha irmã, Amanda, pela amizade, torcida e por estar sempre ao meu lado. Amo vocês.

À minha orientadora, Professora Silvia Lopes de Sena Taglialenha, minha profunda gratidão. Sua paciência, conhecimento e direcionamento preciso foram fundamentais para o desenvolvimento e a conclusão deste trabalho. Obrigada por acreditar neste projeto e por me guiar com tanta competência.

Minha profunda gratidão também à oportunidade de estagiar no Departamento de Trânsito de Joinville. Agradeço às minhas coordenadoras, Nathalia e Leticia, por todos os ensinamentos e pela paciência. À minha parceira Hellen, pela incrível sintonia e ajuda mútua. Aos meus gerentes, Samuel e Cesar, que desde o início confiaram em meu trabalho e me incentivaram a crescer. Aos queridos amigos que a rotina me deu, Netto, Gilson, Karla, Rogério e Costódio, obrigado pelas risadas e pela amizade.

Aos meus amigos de longa data, Karolaine e Patrick, que trilharam comigo toda a jornada da graduação. Passamos por tudo juntos, e ter a amizade de vocês tornou o caminho muito mais fácil e divertido.

Por fim, agradeço ao meu namorado, Emerson. Sua presença foi meu grande diferencial, especialmente nesta reta final tão desafiadora. Obrigada por ser meu suporte, por acalmar minhas ansiedades e por me dar forças quando eu mais precisei. Sem você, certamente teria sido mais difícil. Meu muito obrigada por tudo.

RESUMO

A pintura de sinalizações horizontais desempenha papel fundamental na organização do tráfego e na segurança viária. No município de Joinville, o processo de programação das ordens de serviço do Departamento de Trânsito (DETRANS) ainda é realizado de forma manual e empírica, o que resulta em rotas pouco eficientes, altos custos operacionais e resposta lenta às demandas da comunidade. Diante desse cenário, esta pesquisa propõe um modelo de otimização para o planejamento das atividades de pintura horizontal, com foco na designação e roteirização das equipes de campo. O problema, denominado Problema Integrado de Designação, Roteirização e Escalonamento de Serviços (PIDRES), foi modelado inicialmente por meio de um modelo de Programação Linear Inteira Mista (PLIM), adequado para instâncias de pequeno porte. Para instâncias maiores, foram aplicadas as meta-heurísticas Variable Neighborhood Descent (VND) e Variable Neighborhood Search (VNS), permitindo explorar o espaço de busca de forma eficiente e superar limitações de complexidade computacional. Os resultados experimentais confirmaram os limites computacionais do PLIM, que obteve soluções ótimas apenas para instâncias com até 12 tarefas. Para instâncias maiores, a meta-heurística VNS demonstrou desempenho consistentemente superior ao VND, sendo capaz de escapar de ótimos locais e gerar soluções de alta qualidade (inclusive a ótima em alguns casos) em tempos computacionais significativamente inferiores aos do método exato. Análises indicaram que cerca de 100 iterações do VNS oferecem um bom equilíbrio entre esforço computacional e qualidade da solução. A metodologia proposta constitui, portanto, uma ferramenta analítica robusta aplicável ao contexto real do DETRANS, com potencial para apoiar a tomada de decisão, otimizar a alocação de recursos e melhorar a eficiência operacional do serviço de pintura em Joinville.

Palavras-chave: programação da pintura de sinalizações horizontais de vias urbanas; alocação de ordens de serviço; roteirização, scheduling.

ABSTRACT

Horizontal road markings play a fundamental role in traffic organization and road safety. In the municipality of Joinville, the process of scheduling service orders by the Department of Transit (DETRANS) is still carried out manually and empirically, resulting in inefficient routes, high operational costs, and slow response to community demands. Given this scenario, this research proposes an optimization model for planning horizontal painting activities, focusing on the assignment and routing of field teams. The problem, termed the Problem of Integrated Designation, Routing, and Scheduling of Services (PIDRES), was initially modeled using a Mixed Integer Linear Programming (MILP) model, suitable for small-sized instances. For larger instances, the metaheuristics Variable Neighborhood Descent (VND) and Variable Neighborhood Search (VNS) were applied, allowing for efficient exploration of the search space and overcoming computational complexity limitations.

Experimental results confirmed the computational limits of the MILP model, which obtained optimal solutions only for instances with up to 12 tasks. For larger instances, the VNS metaheuristic demonstrated consistently superior performance compared to VND, managing to escape local optima and generate high-quality solutions (including the optimal one in some cases) in significantly lower computational times than the exact method. Analyses indicated that around 100 VNS iterations offer a good balance between computational effort and solution quality. The proposed methodology therefore constitutes a robust analytical tool applicable to the real context of DETRANS, with the potential to support decision-making, optimize resource allocation, and improve the operational efficiency of the painting service in Joinville.

Keywords: road marking painting; service order allocation; routing; scheduling.

LISTA DE SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
TCC	Trabalho de Conclusão de Curso
VND	Variable Neighborhood Descent
VNS	Variable Neighborhood Search
PLIM	Programação Linear Inteira Mista
CET	Companhia de Engenharia de Tráfego
CONTRAN	Conselho Nacional de Trânsito
SENATRAN	Secretária Nacional de Trânsito
GAP	Generalized Assigment Problem
OS	Ordens de Serviço
FO	Função Objetivo
PIDRES	Problema Integrado de Designação, Roteirização e Escalonamento de Serviços.
AMRs	Autonomous Mobile Robot
PRISMA	Preferred Reporting Items for Systematic reviews and Meta-Analyses
AHP	Processo Analítico Hierárquico

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE CÓDIGOS

SUMÁRIO

1 Introdução

A sinalização viária é um dos pilares fundamentais da segurança e da organização do trânsito urbano. Entre os diversos elementos que compõem essa estrutura, as faixas de sinalização horizontal (linhas divisórias de pista, faixas de pedestres e setas direcionais) desempenham papel essencial na orientação dos condutores e na proteção dos pedestres, orientando e controlando deslocamentos em cenários com desafios de geometria, topografia ou diante de obstáculos. Também complementa os sinais verticais de regulamentação, advertência e indicação, tendo, em casos específicos, caráter regulamentador (Companhia de Engenharia de Tráfego CET, 2019).

A sinalização horizontal transmite informações de forma clara e eficiente, permitindo que condutores e pedestres as compreendam sem desviar a atenção da via. A comunicação precisa das mensagens deve assegurar que todos os usuários, independentemente de sua origem ou familiaridade com o local, as reconheçam e compreendam (SENATRAN, 2022).

A sinalização horizontal apresenta limitações como durabilidade reduzida dos materiais em vias com tráfego intenso, visibilidade prejudicada em situações de congestionamento e comprometimento da visualização quando o pavimento está molhado, devido à reflexão da luz natural ou artificial pela camada de água (Companhia de Engenharia de Tráfego CET, 2019). Esses desafios evidenciam a importância de uma gestão eficiente e de estratégias para aplicação das pinturas viárias, que garantam a segurança e a fluidez do trânsito.

No contexto brasileiro, cidades de médio e grande porte, como Joinville (SC), a manutenção e renovação dessas faixas representam um desafio logístico e operacional significativo, especialmente diante da crescente demanda por mobilidade urbana eficiente e segura.

Com o aumento da frota de veículos, da densidade populacional e da complexidade da malha viária, torna-se cada vez mais necessário que os serviços públicos adotem estratégias inteligentes para planejar e executar ações de infraestrutura urbana. A pintura de faixas, embora aparentemente simples, envolve uma série de variáveis que impactam diretamente sua eficácia: tempo de execução, condições climáticas, fluxo de veículos, disponibilidade de equipes, custo dos insumos e priorização de trechos críticos. A ausência de um planejamento otimizado pode resultar

em desperdício de recursos, atrasos na execução e comprometimento da segurança viária.

A alocação de equipes demanda um planejamento contínuo por parte dos responsáveis pelo setor e pelos recursos disponíveis. Esse planejamento tem como objetivos: a programação das equipes, a alocação e escalonamento, o treinamento, a especialização, o aprimoramento e a contratação de recursos, assim como a reprogramação devido a necessidades dos clientes, imprevistos com recursos e possíveis atrasos, entre outros aspectos (Azevedo, 2010).

Neste contexto, a otimização da pintura de faixas de sinalização horizontal surge como uma alternativa estratégica para melhorar a eficiência dos serviços prestados, reduzir custos operacionais e garantir maior cobertura e qualidade na sinalização urbana.

A necessidade de desenvolver um modelo para otimizar o serviço de alocação de recursos às tarefas no Detrans Joinville, surgiu devido à dificuldade de se criar uma alocação que maximize o aproveitamento do atendimento do maior número possível de tarefas. A eficiência no planejamento e execução dessas atividades é essencial, uma vez que a sinalização de qualidade contribui diretamente para a segurança viária, a organização do tráfego e a redução de acidentes (Senatran - Secretaria Nacional de Trânsito Contran, 2022).

Atualmente, o processo de alocação de tarefas no Departamento de Trânsito de Joinville, baseado em práticas manuais e decisões pontuais, gera ineficiências que comprometem a utilização ideal dos recursos e aumentam os custos operacionais. A alocação inadequada das equipes tem resultado em desperdícios e em altos custos operacionais, além de comprometer a qualidade da sinalização viária (Senatran - Secretaria Nacional de Trânsito Contran, 2022).

Diante deste cenário, desenvolveu-se um modelo de Programação Linear Inteira Mista (PLIM), denominado Problema Integrado de Designação, Roteirização e Escalonamento de Serviços (PIDRES), para tratar instâncias de pequeno porte. O modelo proposto determina a designação, a programação e a roteirização das rotas para realização dos serviços de pintura de sinalização horizontal em vias urbanas no município de Joinville, de modo a minimizar uma função objetivo que pondera o custo de deslocamento entre as tarefas, o valor bônus gerado pela execução das ordens de serviço e a penalidade por atrasos.

Para tratar instâncias de grande porte, devido à complexidade do PIDRES, foram propostos métodos de soluções aproximados, baseados nas meta-heurísticas Variable Neighborhood Descent (VND) e Variable Neighborhood Search (VNS), que exploram o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança durante o processo de busca local (Mladenovic e Hansen, 1997). Os testes computacionais foram conduzidos com o auxílio de ferramentas de otimização

implementadas em Python.

1.1 OBJETIVOS

Foram definidos os objetivos a seguir.

1.1.1 Objetivo Geral

Desenvolver e aplicar um modelo de otimização para o planejamento logístico do processo de pintura de sinalização horizontal em vias urbanas no Departamento de Trânsito do município de Joinville.

1.1.2 Objetivos específicos

- Identificar as práticas atuais de alocação de ordens de serviço de pintura no Detrans Joinville;
- Identificar e mapear as restrições operacionais para a modelagem;
- Apresentar um modelo matemático que represente o processo de alocação das ordens de serviço;
- Aplicar o modelo de designação, sequênciamento (scheduling) e roteirização, para identificar as combinações ótimas de alocação de tarefas;
- Implementar uma Heurística Construtiva Gulosa para o PIDRES, com o objetivo de gerar rapidamente soluções iniciais viáveis;
- Aplicar e desenvolver as meta-heurísticas baseadas em Variable Neighborhood Search (VNS) e Variable Neighborhood Descent (VND) a partir das soluções iniciais obtidas pela heurística gulosa, visando aprimorar a qualidade das soluções;
- Realizar testes computacionais para validação e análise de desempenho, comparando os resultados do método exato (PLIM) e das meta-heurísticas (VND/VNS);
- Analisar o impacto do número de iterações do VNS, na eficiência do método proposto em termos de melhoria da solução versus tempo de execução;
- Propor melhorias e recomendações para a execução do serviço com base nos resultados obtidos;

1.2 RESULTADOS ESPERADOS

- Um modelo de otimização capaz de gerar rotas e cronogramas mais eficientes, reduzindo custos operacionais e tempo de execução;
- Ferramentas e diretrizes para subsidiar o planejamento e a tomada de decisão pela gestão municipal;

- Potencial replicação da metodologia para outros municípios de porte semelhante.

1.3 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado em seis capítulos, incluindo esta introdução, estruturados de forma a guiar o leitor desde a contextualização do problema até a apresentação e discussão dos resultados.

No Capítulo 2 apresenta-se a metodologia da pesquisa, indicando as etapas seguidas para a formulação do problema, o desenvolvimento das soluções e a análise dos resultados.

No Capítulo 3 é apresentada a fundamentação teórica, abordando os conceitos de otimização combinatória, os problemas clássicos que compõem o PIDRES (Designação, Roteirização e Escalonamento) e as meta-heurísticas VND e VNS.

No Capítulo 4 são detalhados os materiais e métodos, incluindo a coleta e tratamento de dados, a implementação computacional do modelo PLIM e o desenvolvimento dos algoritmos heurísticos.

Os resultados obtidos com a aplicação dos métodos exato e aproximado são apresentados e discutidos no Capítulo 5, com foco na análise comparativa de desempenho, qualidade da solução e tempo computacional.

Por fim, no Capítulo 6 apresenta-se a conclusão do trabalho, destacando-se as principais contribuições, o atendimento aos objetivos e sugestões para trabalhos futuros.

2 Metodologia

No presente capítulo descreve-se a estratégia metodológica adotada para o desenvolvimento do trabalho, contemplando tanto a classificação da pesquisa quanto os procedimentos de coleta, modelagem e análise dos dados.

Considerando que o objetivo central consiste em desenvolver e aplicar um modelo de otimização para a programação de ordens de serviços de pintura de sinalização horizontal em Joinville/SC, optou-se por uma abordagem que integra pesquisa bibliográfica sistemática, diagnóstico da realidade operacional do município, formulação matemática do problema, implementação computacional de métodos de solução (exatos e heurísticos) e de análise comparativa de cenários simulados.

A revisão bibliográfica foi estruturada segundo a metodologia PRISMA, que permite a seleção criteriosa e transparente das referências, garantindo abrangência e confiabilidade na base teórica. O diagnóstico da situação atual foi realizado por meio do levantamento de dados junto ao Departamento de Trânsito de Joinville, englobando informações sobre ordens de serviço, equipes disponíveis e restrições operacionais.

Na etapa de modelagem, o problema foi estruturado combinando três abordagens clássicas da Pesquisa Operacional: o Problema de Designação, focado em atribuir qual equipe de pintura executará quais ordens de serviço; O Problema de Escalonamento em Máquinas Paralelas, que define a sequência de execução dessas ordens em cada equipe; e o Problema de Roteirização, que determina a ordem ótima de visita às ordens de serviço alocadas a cada equipe, minimizando o tempo total de deslocamento. Neste contexto, as equipes atuam como recursos (máquinas) e as ordens de serviço como tarefas. Para a resolução, foram implementadas em Python as metaheurísticas Variable Neighborhood Descent (VND) e Variable Neighborhood Search (VNS).

Por fim, foram realizados testes computacionais para avaliar o desempenho do modelo de otimização e das meta-heurísticas propostas, considerando indicadores de qualidade da solução, custo e tempo. Os resultados obtidos embasam a análise crítica e a formulação de recomendações práticas para a gestão da sinalização viária no município.

A seguir, apresenta-se inicialmente a classificação da pesquisa, que detalha sua natureza, abordagem e procedimentos. Em seguida, descrevem-se as etapas metodológicas, que organizam o desenvolvimento do trabalho desde a revisão

bibliográfica até a análise dos resultados.

2.1 CLASSIFICAÇÃO DA PESQUISA

- Natureza: A pesquisa é classificada como aplicada, uma vez que busca gerar conhecimentos voltados à solução prática de um problema específico. Conforme explicam Marconi e Lakatos (2003), esse tipo de pesquisa visa contribuir diretamente para a resolução de demandas concretas. Nesse contexto, o trabalho desenvolve um modelo matemático aplicável ao planejamento dos serviços de pintura viária da Prefeitura de Joinville.
- Abordagem: A abordagem metodológica é quantitativa, pois fundamenta-se na mensuração e análise de dados numéricos. De acordo com Vergara 2009, a pesquisa quantitativa caracteriza-se pelo uso de instrumentos formais e procedimentos estatísticos ou matemáticos para interpretação dos dados. No presente estudo, essa abordagem manifesta-se na formulação de um modelo de otimização e na avaliação de seu desempenho por meio de indicadores objetivos, como o valor da função objetivo e o tempo computacional de execução.
- Objetivos: A pesquisa apresenta um duplo caráter, sendo ao mesmo tempo descritiva e propositiva.
  - Descritiva: na fase inicial, descrevem-se as características do processo de planejamento das equipes de pintura em Joinville, em conformidade com a definição de Gil (2002) sobre pesquisas que buscam registrar, analisar e correlacionar fatos sem manipulá-los.
  - Propositiva: o foco central é a proposição de um novo artefato, representado pelo modelo de otimização matemática, estruturado para apoiar a tomada de decisão e aumentar a eficiência operacional.
- Quanto aos Procedimentos Técnicos: O trabalho combina o estudo de caso com a pesquisa experimental. O estudo de caso, conforme Gil (2002), consiste na análise aprofundada de uma situação concreta, neste caso, a operação de serviços de sinalização da Prefeitura de Joinville. Já a pesquisa experimental manifesta-se na aplicação computacional do modelo proposto, submetido a testes em diferentes cenários, de modo a avaliar sua eficácia e robustez.

2.2 ETAPAS METODOLÓGICAS

A condução desta pesquisa seguiu um conjunto estruturado de etapas metodológicas, organizadas de forma sequencial e interdependente, ilustradas na Figura 1. O detalhamento dessas etapas é apresentado na sequência, e permite compreender a lógica de desenvolvimento do trabalho e a fundamentação que sustenta cada decisão metodológica.

Figura 1 – Etapas metodológicas

Fonte: Autor (2025)

2.2.1 Revisão Bibliográfica

Consiste em um levantamento teórico sobre problemas clássicos de otimização combinatória que formam a base do PIDRES. O estudo focou no Problema de Designação Generalizado, que fundamenta a designação das ordens de serviço às equipes; no Problema de Escalonamento de Tarefas em Máquinas Paralelas, relacionado ao sequenciamento da execução em cada equipe; e no Problema de Roteirização, que aborda a definição das rotas ótimas. Também foram abordados métodos aproximados, como heurísticas construtivas para geração de soluções iniciais e as meta-heurísticas VND e VNS como técnicas eficazes para o refinamento de soluções em problemas de grande porte.

2.2.2 Identificação do Problema

Nesta seção define-se formalmente o Problema Integrado de Designação, Roteirização e Escalonamento de Serviços (PIDRES), proposto neste trabalho.

Ressalta-se que PIDRES pode ser decomposto em problemas clássicos da literatura: o Problema de Designação (Assignment Problem) (Arenales et al., 2007), o Problema de Escalonamento em Máquinas Paralelas (Parallel Machine Scheduling) (Pinedo, 2016) e o Problema de Roteirização de Veículos (PRV)(Toth e Vigo, 2014).

2.2.3 Coleta e Tratamento de Dados

Os dados operacionais fornecidos pelo Detrans são coletados e tratados. As informações essenciais (localizações, áreas, prioridades das Ordens de Serviço) são organizadas para se converterem em parâmetros do modelo. A etapa inclui o cálculo dos tempos de deslocamento e de execução das tarefas, elementos indispensáveis para a modelagem precisa do problema.

2.2.4 Métodos de Solução e Otimização Matemática

Esta etapa envolve a estruturação do problema como um modelo de Programação Linear Inteira Mista (PLIM). A formulação contempla a definição detalhada dos parâmetros (dados de entrada), das variáveis de decisão, das restrições e da Função Objetivo, definida para minimizar penalidades associadas a atrasos, por exemplo. Esta estruturação visa garantir a fidelidade do modelo ao problema real.

2.2.5 Implementação Computacional e Resolução

Nesta etapa, o modelo PLIM é implementado em Python utilizando o solver Gurobi para resolução. O modelo exato é aplicado, inicialmente, em instâncias de menor porte para validar a formulação e avaliar seus limites de tratabilidade computacional.

Para lidar com instâncias maiores, adota-se uma abordagem complementar: primeiro, uma heurística construtiva (gulosa) gera uma solução inicial; em seguida, as meta-heurísticas Variable Neighborhood Descent (VND) e Variable Neighborhood Search (VNS) são aplicadas para refinar essa solução, buscando resultados próximos ao ótimo em tempo de execução viável.

2.2.6 Análise dos Resultados e Discussões

Nesta etapa, realiza-se a avaliação da metodologia por meio de uma análise comparativa. A performance do modelo matemático exato é examinada para determinar seu limite de aplicação prática, identificando a maior dimensão de problema solucionável em tempo razoável. A eficácia da abordagem heurística é medida pela capacidade

de reproduzir soluções ótimas em instâncias menores e pelo seu desempenho em instâncias maiores. Por fim, a discussão pondera a relação custo-benefício entre os métodos, destacando o equilíbrio entre a otimalidade e a agilidade computacional.

2.2.7 Conclusões

Consolidam-se os principais achados da pesquisa, destacando as contribuições mais relevantes do estudo. Discutem-se de forma crítica as limitações e potencialidades de cada abordagem, bem como sua adequação a diferentes cenários práticos. Por fim, são apresentadas as implicações do modelo e das heurísticas para a gestão operacional, além de sugestões para trabalhos futuros.

3 Fundamentação teórica

O presente capítulo dedica-se à construção da base teórica que sustenta esta pesquisa. Para fundamentar a solução proposta para o problema de planejamento de equipes em Joinville, realiza-se aqui uma revisão sobre os pilares da Pesquisa Operacional e da otimização, seguida de uma análise de problemas análogos na literatura e uma discussão sobre algumas metodologias de solução existentes. A finalidade é estabelecer a base conceitual sobre a qual o modelo matemático deste trabalho é formulado e validado.

A análise da literatura será, portanto, direcionada para abordagens focadas na eficiência. A investigação buscará identificar métodos que ofereçam um balanço prático entre a qualidade da solução e o tempo computacional necessário para obtê-la. Com isso, almeja-se que esta fundamentação não apenas justifique o modelo proposto, mas também guie a escolha de uma metodologia de resolução que seja compatível com as demandas práticas do problema.

Este trabalho trata de um problema real de logística e planejamento, focado na prestação de serviços para a Prefeitura de Joinville. O desafio é organizar o trabalho diário de uma empresa terceirizada que realiza os serviços de pintura na cidade. A Prefeitura cria as Ordens de Serviço (OS), e a empresa precisa decidir qual a melhor forma de usar suas equipes para atender a essas ordens, que estão espalhadas pela cidade e têm diferentes durações e níveis de prioridade.

Esse tipo de problema, que precisa decidir as rotas para as equipes e ao mesmo tempo respeitar limites como a jornada de trabalho, é conhecido por ser muito complexo na área de Pesquisa Operacional. Autores clássicos como Hillier e Lieberman (2013) explicam que a dificuldade vem do número gigantesco de combinações possíveis de rotas e alocações de tarefas. Por causa disso, tentar planejar tudo manualmente ou por intuição quase nunca leva à melhor solução, o que justifica o uso de ferramentas de apoio mais estruturadas.

Para resolver essa questão, a abordagem escolhida foi a criação de um modelo de otimização matemática. O uso de modelos matemáticos é de grande interesse para as organizações, pois, como destaca Ballou (2006), eles ajudam a reduzir custos sem prejudicar a qualidade do serviço. O objetivo principal deste trabalho é, portanto, desenvolver um modelo que funcione como uma ferramenta prática. Ele deverá gerar um plano de trabalho diário que diga quais OS atender, qual equipe irá e a sequência

a ser seguida, buscando sempre o melhor equilíbrio entre atender as prioridades da cidade e minimizar os custos com deslocamento.

Sem uma ferramenta de otimização, o planejamento das equipes pode ser ineficiente e gerar custos desnecessários para o serviço público. Isso pode causar rotas mais longas, maior gasto com combustível e sobrecarga das equipes. Consequentemente, serviços importantes para a população podem atrasar, gerando insatisfação. Dessa forma, este trabalho se justifica pela necessidade de criar uma solução mais inteligente e eficiente para o planejamento desses serviços, ajudando a usar melhor os recursos e a melhorar a qualidade do atendimento prestado à comunidade de Joinville.

3.1 PESQUISA OPERACIONAL (PO)

Pesquisa Operacional é um método científico de tomada de decisões que consiste na descrição de um sistema organizado com o auxílio de modelo e, através da experimentação com o modelo, na descoberta da melhor maneira de operar o sistema (Silva et al., 1998). Para Costa (1997), PO é a aplicação do método científico, por equipes interdisciplinares, a problemas que dizem respeito ao controle de sistemas organizados, com o propósito de obter as soluções que satisfaçam aos objetivos da organização.

Três requisitos são necessários para a utilização da PO. O primeiro envolve a compreensão de características e atributos de um sistema complexo e a habilidade de abstrair e traduzir os pontos importantes em um modelo matemático ou de simulação. O segundo consiste na habilidade para desenvolver métodos de resolução para os modelos e utilizar pacotes comerciais com conhecimento sobre os métodos utilizados nestes. O terceiro envolve a comunicação com clientes para compreender o problema e explicar resultados não intuitivos, mas importantes, gerados pela aplicação de pesquisa operacional (Arenales et al., 2007).

Na Pesquisa Operacional, busca-se a configuração ótima de um sistema, no entanto, essa otimização depende de um modelo matemático adequado; se o modelo não for desenvolvido corretamente, a solução resultante pode não ser viável na prática. Quanto mais robusto for o modelo matemático, menor será a probabilidade de a solução ótima ser inviável na realidade, por essa razão, a modelagem é a etapa que exige maior raciocínio e é a única fase da resolução de um problema de PO que não pode ser realizada com o auxílio de um computador (Costa, 1997).

Segundo Silva et al. (1998), a modelagem de um problema na PO costuma envolver seis passos:

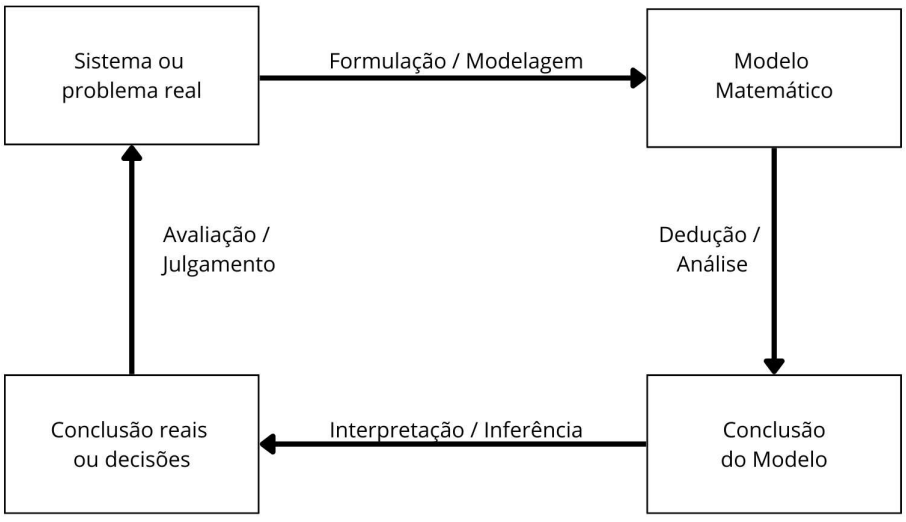
1. Formulação do problema;
2. Construção do modelo do sistema;

3. Cálculo da Solução através do modelo;
4. Teste do modelo e da solução;
5. Estabelecimento de controles da solução;
6. Implantação e acompanhamento.

A programação matemática lida com problemas de decisão utilizando modelos matemáticos que representam o problema real. Nesses modelos, as variáveis são definidas e as relações matemáticas são estabelecidas de acordo com os objetivos e restrições do problema, para descrever o comportamento do sistema. A resolução do modelo fornece soluções que, baseadas nos dados do problema, devem ser validadas para verificar sua compatibilidade com a realidade (Arenales et al., 2007).

Embora essas soluções apoiem a tomada de decisões, outros fatores intangíveis também devem ser considerados para a decisão final, como experiência dos gestores, contexto social e prioridades políticas. É importante ressaltar que modelos matemáticos não substituem os tomadores de decisão, mas oferecem suporte valioso para decisões mais informadas (Rivas, 2009). Na Figura 2 apresenta-se uma representação do processo de modelagem utilizado para modelos de otimização e tomada de decisão.

Figura 2 – Processo de Modelagem



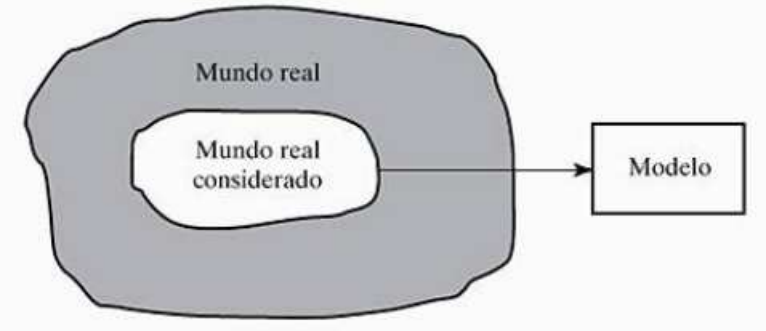
Fonte: Adaptado de Arenales (2007, p. 4).

A confiabilidade da solução obtida pelo modelo depende de sua validação na representação do sistema real. Esse processo consiste em confirmar se o modelo reflete com precisão o comportamento do sistema (Figura 3). Assim, a diferença entre a solução real e a proposta pelo modelo está diretamente relacionada à sua capacidade

de descrever o funcionamento do sistema (Silva et al., 1998).

Problemas simples podem ser representados por modelos igualmente simples, o que facilita a solução. Por outro lado, problemas mais complexos exigem modelos mais elaborados, cuja resolução pode ser consideravelmente mais desafiadora (Lisboa, 2002).

Figura 3 – Níveis de abstração no desenvolvimento do modelo



Fonte: Taha (2008, p. 3)

Recomenda-se o uso de modelos, pois são mais econômicos do que replicar a estrutura real e testar todas as possíveis soluções. Além disso, utilizar um modelo representa um risco menor do que experimentar o impacto de cada alternativa de solução, realizando testes e ajustes no mundo real. Também se economiza tempo, pois o software, com o auxílio das capacidades computacionais, consegue realizar milhares de cálculos em frações de segundo, permitindo obter rapidamente a solução para os problemas. Por outro lado, se quiséssemos testar o impacto de diversas soluções, levaríamos muito tempo apenas para configurar o sistema de acordo com a solução desejada (Garcia e Brando, 2015).

3.1.1 O Problema de Designação

O problema da designação é um tipo especial de problema de programação linear em que os designados estão sendo indicados para a realização de tarefas (Hillier e Lieberman, 2013). Por exemplo, os designados poderiam ser empregados que precisem receber designações de trabalho. De acordo com Hillier (2013) designar pessoas para determinadas tarefas é uma aplicação comum do problema da designação. Entretanto, os designados não precisam ser necessariamente pessoas. Eles também podem ser máquinas, veículos ou fábricas, ou até mesmo períodos a serem destinados a tarefas.

3.1.1.1 Designação Simples

O Problema de Designação, em sua versão clássica, consiste em atribuir um número n de agentes a um mesmo número n de tarefas de forma exclusiva, buscando

minimizar o custo total c_{ij} da operação. Este modelo um-para-um, onde cada agente executa uma única tarefa, é a base teórica para muitos problemas de otimização (Arenales et al., 2007). Definindo-se as variáveis:

$$x_{ij} = \begin{cases} 1 & \text{se a tarefa } i \text{ é designada ao agente } j \\ 0 & \text{caso contrário} \end{cases}$$

o modelo de PLI que representa o problema é definido em (1)-(4).

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{1}$$

$$\text{s.a.} \quad \sum_{i=1}^{n} x_{ij} = 1, \qquad j = 1,\ldots,n \tag{2}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1,\ldots,n \tag{3}$$

$$x_{ij} \in \{0,1\}, \qquad \forall i,j \in \{1,\ldots,n\} \tag{4}$$

A Função Objetivo definida em (1) minimiza o custo total de designação de tarefas a agentes. As restrições (2) e (3) asseguram que cada tarefa j é designada a um único agente, e cada agente i executa exatamente uma tarefa. A restrição (4) indica o tipo das variáveis.

3.1.1.2 Designação Generalizada

Neste problema, tem-se m tarefas e n agentes, com $m > n$; cada tarefa deve ser executada por um único agente, e cada agente pode executar mais de uma tarefa.

A execução da tarefa i pelo agente j requer uma quantidade a_{ij} de recurso do agente j, com custo c_{ij}. O agente j tem capacidade de recurso b_j (Arenales et al., 2007).

Definindo-se as variáveis binárias x_{ij}, que indicam se a tarefa i é ou não atribuída ao agente j, obtém-se o seguinte modelo:

$$\min \quad \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{5}$$

$$\text{s.a.} \quad \sum_{i=1}^{m} x_{ij} = 1, \qquad \forall j \in \{1,\ldots,n\} \tag{6}$$

$$\sum_{j=1}^{n} a_{ij}x_{ij} \leq b_j, \qquad \forall i \in \{1,\ldots,m\} \tag{7}$$

$$x_{ij} \in \{0,1\}, \qquad \forall i \in \{1,\ldots,m\}, \forall j \in \{1,\ldots,n\} \tag{8}$$

A Função Objetivo definida em (5) minimiza o custo total da designação de tarefas aos agentes. As restrições de igualdade (6) garantem que cada tarefa i é executada por um único agente j, e as restrições de desigualdade (7) impõem que a capacidade b_j de cada agente i não é excedida. A restrição (8) define o domínio das variáveis.

3.1.2 Scheduling

O problema de sequenciamento ou de programação (*scheduling problem*) consiste em um processo de otimização focado na alocação de recursos como máquinas, ferramentas e mão de obra. Essa otimização, conforme destaca (Baker, 1974), tem como finalidade a máxima utilização dos recursos disponíveis dentro de um horizonte de tempo específico para a execução de um conjunto de tarefas.

De forma complementar, (Reis, 2006) detalha que os problemas de *scheduling* geralmente assumem a necessidade de processar um determinado número de operações, definindo a sequência em que serão executadas. O autor ressalta duas condições centrais: cada operação necessita de um recurso (uma máquina ou operador) por um intervalo de tempo definido, e cada recurso pode executar apenas uma operação por vez.

Para aprofundar a compreensão do tema, (Pinedo, 2016) define alguns dos conceitos essenciais para a modelagem do problema:

- Tempo de processamento: Período que um recurso leva para realizar uma operação.
- Tempo de setup: Tempo de preparação do recurso para uma nova operação.
- Tempo de conclusão (makespan): Momento em que a última tarefa do conjunto é finalizada.
- Restrição: Qualquer condição que limite as possibilidades de escolha.
- Produto: Resultado de um processo fabril.
- Prioridade/Peso: Característica que determina a urgência de uma tarefa em relação a outra.

- Data de disponibilidade: Momento a partir do qual um recurso está livre para processamento.
- Recursos: Máquinas, matérias-primas, mão de obra ou outros elementos necessários.
- Tarefa (job): Um conjunto de operações a serem processadas.

O problema de sequenciamento, ou *scheduling*, é fundamental na tomada de decisão em sistemas produtivos. Para (Branquinho, 2013), a conclusão de uma tarefa necessita da realização de várias operações em uma determinada ordem. Assim, define-se o problema de sequenciamento como um problema de agendamento de operações, em um determinado número de máquinas com suas respectivas características, como tempos de processamento e tempos de *setup*, visando buscar o menor tempo possível da execução total de todas as tarefas (*makespan*).

Segundo (Pape, 2015), o problema de *scheduling* pode ser dividido da seguinte maneira:

- Designação: Define quem irá realizar cada tarefa;
- Sequenciamento: Define qual será a ordem de realização de cada tarefa;
- Agendamento ou escalonamento: Determina quando será realizada cada tarefa (quando inicia e quando finaliza) satisfazendo todas as restrições exigidas no problema.

Por ser um problema importante na tomada de decisão, o problema de *scheduling* pode compreender diferentes critérios de desempenho do sistema. Segundo (Reklaitis, 1982), alguns critérios de desempenho de sistema podem ser:

- *Makespan:* Tempo total para o processamento de todas as tarefas;
- *Mean flow-time:* Média de tempo que dura o fluxo de tarefas em todo o sistema;
- *Total flow-time:* Soma do tempo total de duração do fluxo de tarefas;
- *Mean tardiness:* Atraso máximo considerado para a conclusão de tarefas;
- *Tardiness:* Soma das penalidades devido ao atraso;
- *Earliness:* Soma das penalidades pelo adiantamento.

Segundo (Hochbaum, 1999), os problemas de sequenciamento têm os seguintes objetivos:

- Minimizar o tempo de execução total das tarefas, analisando os níveis de utilização das máquinas;
- Minimizar o tempo de espera de cada tarefa (o tempo entre o término de uma tarefa e o começo de outra);
- Minimizar os custos de execução de cada atividade.

De acordo com (Loureiro, 2014), o problema de *scheduling* tem forte aplicação na área de planejamento e controle da produção (PCP), e de modo geral tem como objetivo definir os tempos de início e de finalização que um conjunto de tarefas devem ser processadas por um conjunto de recursos, de forma a otimizar uma medida de desempenho relacionada ao tempo. Para (Joo e Kim, 2015), a etapa de *scheduling* é de grande impacto para o planejamento e controle da produção, por se tratar da programação de uma série de atividades que tornam disponíveis os recursos necessários para a conclusão das tarefas, determinando período de início para a produção.

Para (Fernandes, 2021), o modo em que a linha de produção está funcionando pode afetar a ordem dos processos, logo o problema de sequenciamento de tarefas está relacionado à forma em que está o posicionamento de máquinas. Portanto, a seguir será apresentado problema de sequenciamento em máquinas e seus respectivos modelos.

3.1.2.1 Modelo de Uma Única Máquina

De acordo com (Pinedo, 2012), o Modelo de Máquina Única é o mais simples entre os existentes. Ele se caracteriza pelo processamento de todos os produtos em uma única máquina especializada, que é considerada como um recurso único - seja um equipamento, uma célula de produção ou um conjunto de recursos modelados como um só. Um esquema representativo deste modelo pode ser visualizado na Figura 4.

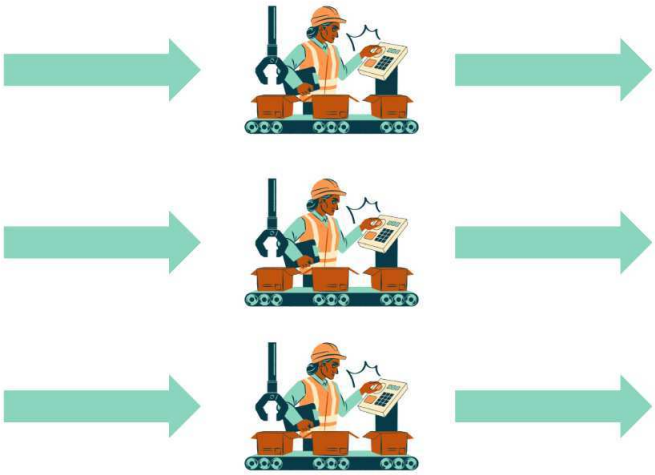
Figura 4 – Representação de processamento em máquina única

Fonte: Autor (2025)

3.1.2.2 Modelo de Máquinas em Paralelo

O Modelo de Máquinas em Paralelo, já introduzido na seção anterior, utiliza múltiplos recursos para executar a mesma operação. Uma distinção crucial, apontada por Pinedo (2008), é que os tempos de processamento entre as máquinas podem variar. No cenário de máquinas idênticas, qualquer uma das m máquinas pode processar os produtos sem impactar o makespan. Em contrapartida, quando as máquinas possuem velocidades distintas, a eficiência de cada uma afeta diretamente o tempo total de produção, visto que o modelo tenderá a alocar mais tarefas à máquina com o menor tempo de processamento. Um esquema deste modelo é ilustrado na Figura 5.

Figura 5 – Representação de processamento em máquinas paralelas



Fonte: Autor (2025)

Na Figura 5 esquematiza-se uma configuração onde múltiplos produtos são alocados em diversas máquinas que operam em paralelo. Essa estrutura permite o processamento simultâneo dos itens, que ocorrem concorrentemente até a sua saída do sistema produtivo.

3.1.3 O Problema de Roteirização em Veículos (PRV)

O Problema de Roteirização de Veículos (PRV) é uma área fundamental da Pesquisa Operacional, crucial para o planejamento logístico estratégico (Assad, 1988). Seu objetivo principal é determinar rotas de menor custo total para atender a um conjunto pré-definido de pontos (Belfiore, 2005). Atingir essa meta envolve decisões sobre alocação de recursos e sequenciamento de visitas, respeitando restrições operacionais como capacidade e jornada de trabalho (Novaes, 2004). Dada a diversidade de cenários, os PRVs são categorizados com base em suas funções objetivo, restrições e características específicas (Bodin et al., 1983).

Considerando um conjunto de n clientes com demanda $q_i, i = 1, \ldots, n$ e m veículos de capacidade $Q_k, k = 1, \ldots, m$, a modelagem matemática do PRV pode ser representada pelas Equações (9)-(16) formuladas por Fisher e Jaikumar (Fisher e Jaikumar, 1981), em que c_{ij} é o custo de ir do cliente i ao cliente j, e as variáveis de decisão x_{ijk} e y_{ik} são definidas como:

$$x_{ijk} = \begin{cases} 1, & \text{se veículo } k \text{ vai imediatamente do cliente } i \text{ ao cliente } j \\ 0, & \text{caso contrário} \end{cases}$$

$$y_{ik} = \begin{cases} 1, & \text{se o veículo } k \text{ visita o cliente } i \\ 0, & \text{caso contrário} \end{cases}$$

A formulação matemática é a seguinte:

Função Objetivo:

$$\text{Minimizar } Z = \sum_{i,j} \sum_{k} c_{ij} x_{ijk} \tag{9}$$

Sujeito a:

$$\sum_{k} y_{ik} = 1, \quad \forall i = 2, \ldots, n \tag{10}$$

$$\sum_{k} y_{1k} = m \tag{11}$$

$$\sum_{i} q_i y_{ik} \leq Q_k, \quad \forall k = 1, \ldots, m \tag{12}$$

$$\sum_{j} x_{ijk} = \sum_{j} x_{jik} = y_{ik}, \quad \forall i = 1, \ldots, n, \quad \forall k = 1, \ldots, m \tag{13}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ijk} \leq |S| - 1, \quad \forall S \subseteq \{2, \ldots, n\}, S \neq \emptyset, \quad \forall k = 1, \ldots, m \tag{14}$$

$$y_{ik} \in \{0, 1\}, \quad \forall i = 1, \ldots, n, \quad \forall k = 1, \ldots, m \tag{15}$$

$$x_{ijk} \in \{0, 1\}, \quad \forall i, j = 1, \ldots, n, \quad \forall k = 1, \ldots, m \tag{16}$$

A função objetivo (9) minimiza o custo total. A restrição (10) garante que cada cliente i seja visitado por apenas um veículo k. A restrição (11) garante que o depósito receba uma visita de todos os veículos. A restrição (12) garante que a soma das demandas dos clientes integrantes de cada rota não ultrapasse a capacidade de cada veículo k. A restrição (13) representa restrição de fluxo em rede, que garantem que para cada cliente há apenas uma aresta de entrada e uma de saída associada à visita do veículo k. As restrição (14) assegura que não devem existir sub-rotas e, por fim, as restrições (15) e (16) definem o tipo das variáveis.

3.2 MÉTODOS DE SOLUÇÃO

O propósito desta seção é apresentar e comparar as possíveis estratégias de resolução para o problema de otimização. A seleção de um método apropriado será guiada pela análise de indicadores de desempenho, incluindo a eficiência computacional (tempo de execução), a eficácia da solução (qualidade) e a viabilidade de implementação. A discussão percorre desde as técnicas que buscam a solução ótima, como os algoritmos para Programação Inteira-Mista, até as abordagens aproximadas, notadamente as Heurísticas e Meta-heurísticas, que serão aprofundadas nos tópicos subsequentes.

3.2.1 Métodos exatos

A busca pela solução de um modelo matemático por meio de programação matemática se apoia no uso de métodos exatos. Entre as abordagens mais consolidadas, o Método Simplex é classicamente aplicado a problemas lineares contínuos, enquanto o Método Branch-and-Bound é a técnica fundamental para a resolução de problemas com variáveis inteiras.

Os métodos exatos são projetados para determinar a solução ótima de um modelo, e sua implementação requer o uso de softwares de otimização específicos, conhecidos como solvers, que são escolhidos com base na estrutura do problema. Os avanços computacionais e o surgimento de máquinas de alta performance permitiram que esses métodos se tornassem viáveis para resolver problemas de grande escala do mundo real, que antes eram intratáveis.

No entanto, uma limitação inerente aos métodos de programação matemática é o alto tempo de processamento, especialmente quando aplicados a problemas de sequenciamento, que são de natureza combinatória. Por essa razão, os estudos acadêmicos que utilizam métodos exatos para resolver problemas de sequenciamento de tarefas em máquinas frequentemente se limitam a instâncias de pequeno porte. A formulação desses modelos é comumente realizada em linguagens de modelagem específicas, como AMPL (Fourer, Gay e Kernighan, 2003), que facilitam a integração com solvers computacionais (como LINGO, GUROBI, CPLEX, entre outros) para a resolução de problemas de otimização gerais (ANAND; AGGARWAL; KUMAR, 2017).

Um exemplo da aplicação e limitação dessas técnicas é o trabalho de Che et al. (2017), que propuseram um modelo de Programação Linear Inteira Mista para o problema de sequenciamento em máquinas paralelas não relacionadas com tempos de setup dependentes da sequência. Utilizando o solver CPLEX, os autores conseguiram encontrar soluções ótimas para minimizar o makespan, mas apenas para instâncias de pequeno porte, confirmando a dificuldade computacional de abordagens exatas para problemas combinatórios complexos.

Baldacci *et al* (2006), por exemplo, abordaram um problema de roteirização de veículos que, de forma similar a este trabalho, lida com operações logísticas complexas, especificamente coletas e entregas simultâneas. Os autores desenvolveram um algoritmo exato do tipo Branch-and-Cut e conseguiram resolver otimamente instâncias com até 50 clientes, demonstrando a aplicabilidade do método para problemas de porte moderado.

Já Feillet *et al* (2008) estudaram a otimização da distribuição de jornais em um contexto internacional. Diferentemente do problema de Joinville, o desafio principal era a escala massiva do número de rotas possíveis. Para lidar com essa complexidade, os autores propuseram um método exato mais avançado, o Branch-and-Price, alcançando

soluções ótimas para um problema logístico de grande dimensão que seria intratável por abordagens mais simples.

No contexto da pesquisa brasileira, Ardigo, Luna e Goldbarg (2012) investigaram o problema de roteirização com frota heterogênea, uma característica que adiciona uma camada de decisão similar à escolha de equipes com diferentes capacidades neste trabalho. A solução proposta foi um sofisticado algoritmo híbrido, o Branch-and-Cut-and-Price, demonstrando a aplicação de técnicas de otimização de fronteira para resolver problemas com características realistas e complexas.

Como nos trabalhos anteriormente citados, os métodos de solução via programação matemática, apesar de serem capazes de encontrar a solução ótima, demandam um alto tempo e esforço computacional, especialmente quando se trata de problemas de roteirização e sequenciamento com um elevado número de variáveis e restrições, como o de Joinville. A maneira mais eficaz de contornar esta limitação é com a utilização de heurísticas, que são capazes de encontrar soluções de excelente qualidade em um tempo computacionalmente viável para problemas de grande porte. A seguir, serão explicados em mais detalhes os métodos heurísticos.

3.2.2 Métodos aproximados

Segundo (Taillard, 2023), a abordagem para a resolução de novos problemas frequentemente se baseia em experiências e conhecimentos prévios. No entanto, para desafios de alta complexidade, a busca pela solução ótima global pode ser impraticável. Nesses casos, torna-se aceitável a obtenção de uma solução de alta qualidade, ainda que não seja a melhor possível. Este é precisamente o objetivo dos métodos aproximativos: encontrar uma solução satisfatória dentro de um limite de tempo computacional viável.

3.2.2.1 Métodos Heurísticos

De acordo com Glover e Kochenberger (2003), heurísticas podem ser entendidas como procedimentos de busca projetados para identificar soluções satisfatórias em problemas complexos dentro de um tempo computacional viável, sem a garantia de alcançar a solução ótima global. Trata-se de um conjunto de regras práticas que orientam a exploração do espaço de soluções, permitindo encontrar alternativas de boa qualidade quando os métodos exatos se tornam inviáveis devido à dimensão ou à complexidade do problema.

Um método heurístico é um procedimento que provavelmente vai encontrar uma excelente solução viável, mas não necessariamente uma solução ótima, para o problema específico em questão (Hillier e Lieberman, 2013). Para Arenales et al (2007) heurísticas são métodos de resolução de problemas de otimização discreta

que não garantem a obtenção de uma solução factível ou ótima. Nicholson (1971) propôs uma definição que expressa muito bem as características de uma heurística: é um procedimento para resolver problemas por meio de um enfoque "intuitivo", em geral racional, no qual a estrutura do problema possa ser interpretada e explorada inteligentemente para se obter uma solução razoável.

Uma solução ótima de um problema nem sempre é o alvo dos métodos heurísticos, uma vez que, tendo como ponto de partida uma solução viável, baseiam-se em sucessivas aproximações direcionadas a um ponto ótimo. Logo, estes métodos costumam encontrar as melhores soluções possíveis para problemas, e não soluções exatas, perfeitas, definitivas (Bueno, 2017).

As heurísticas construtivas compreendem uma classe de técnicas que elaboram uma solução final por meio de um processo de agregação incremental. Em cada passo iterativo, o algoritmo avalia e incorpora à estrutura da solução o componente considerado mais vantajoso no contexto local. Um exemplo arquetípico é a heurística gulosa (Applegate et al., 2007)

O principal mérito dessas heurísticas reside em sua eficiência computacional. Contudo, essa vantagem é contrabalanceada por uma desvantagem significativa: a qualidade da solução gerada geralmente não é ótima. A degradação da qualidade ocorre porque a seleção sequencial dos melhores candidatos esgota as boas opções, de modo que os elementos incorporados nas fases finais do processo tendem a impactar negativamente o mérito da solução final.

O paradigma da heurística gulosa consiste em construir uma solução por meio de uma sequência de decisões ótimas locais. Em cada etapa do processo, o algoritmo seleciona a melhor alternativa imediata na esperança de que essa cadeia de escolhas leve a uma solução global de alta qualidade.

No âmbito de problemas de roteirização, essa estratégia é materializada pela Heurística do Vizinho Mais Próximo. De acordo com Taha (2008), seu procedimento parte de um vértice inicial e avança iterativamente para o vértice de menor custo adjacente que ainda não tenha sido inserido na solução. O método prossegue até que um ciclo hamiltoniano seja completado.

Esta heurística, proposta por Bellmore e Nemhauser (1968), é um exemplo de método construtivo que busca a solução final através de ótimos locais (ARENALES et al., 2015). Consequentemente, a solução resultante é, em geral, um ótimo local. A aplicação do algoritmo de Bellmore e Nemhauser (1968) ao problema do caixeiro viajante, conforme Rosenkrantz, Stearns e II (1977), pode ser formalizada nos seguintes passos:

1. Selecionar um vértice de partida arbitrário;
2. Identificar o vértice ainda não visitado mais próximo do último vértice adicionado ao percurso e incluir a aresta que os conecta;

3. Iterar o passo 2 até que todos os vértices estejam no percurso e, por fim, adicionar a aresta que conecta o último vértice ao inicial.

As heurísticas de busca local em vizinhança são amplamente utilizadas em problemas de otimização combinatória, especialmente em contextos nos quais métodos exatos se tornam inviáveis devido à complexidade computacional. O princípio central dessas técnicas é partir de uma solução inicial e explorá-la por meio de pequenas modificações (chamadas de movimentos) que geram soluções vizinhas. A cada iteração, a busca procura identificar melhorias em relação à função objetivo, avançando gradualmente até atingir um ponto no qual não há mais ganhos possíveis (Hillier e Lieberman, 2013).

O conceito de vizinhança é fundamental nesse método e pode ser definido como o conjunto de soluções obtidas por alterações simples, como trocas, inserções ou deslocamentos de elementos em uma solução. A exploração desse espaço pode ocorrer de diferentes formas, resultando em diversas variações de heurísticas locais (Lachtermacher, 2009).

Entre as estratégias mais comuns de busca local em vizinhança destacam-se:

- Busca em vizinhança simples (Hill Climbing): substitui a solução corrente pela melhor encontrada em sua vizinhança imediata, até que não haja mais melhorias (Russell e Norvig, 2013);
- Busca em vizinhança aleatória: seleciona aleatoriamente soluções vizinhas, permitindo escapar de regiões restritas do espaço de busca (Laporte e Osman, 1995);
- Busca em vizinhança adaptativa: ajusta dinamicamente o tamanho ou o tipo da vizinhança conforme o progresso da busca, permitindo maior diversidade na exploração (Mladenovic e Hansen, 1997).

Apesar de sua simplicidade e eficiência, as heurísticas de busca local em vizinhança podem convergir para ótimos locais. Para mitigar essa limitação, frequentemente são combinadas com metaheurísticas mais robustas, como Simulated Annealing e Tabu Search, ou aplicadas em estratégias específicas de variação de vizinhança, como a Variable Neighborhood Descent) e a VNS (Variable Neighborhood Search) (Mladenovic e Hansen, 1997), melhor explicado na próxima seção.

Em (Kailer e Taglialenha, 2023) apresenta-se um estudo sobre o Problema de Designação e Programação de Paletes com Dois Robôs (TRPASP), que busca minimizar o tempo total de coleta e entrega em sistemas com dois robôs operando em lados opostos de um trilho. Um modelo PLIM é utilizado para instâncias pequenas, e uma heurística validada é aplicada a casos maiores, obtendo resultados satisfatórios em tempo reduzido.

3.3 META-HEURÍSTICAS

Uma meta-heurística é um método de resolução geral que fornece tanto uma estrutura quanto diretrizes de estratégia gerais para desenvolver um método heurístico específico que se ajuste a um tipo de problema particular. A meta-heurística se tomou uma das mais importantes técnicas na caixa de ferramentas dos profissionais da PO (Hillier e Lieberman, 2013). A seguir será apresentado os dois métodos metaheurísticos que serão utilizados neste trabalho.

3.3.1 Variable Neighborhood Descent (VND)

O Método de Descida em Vizinhança Variável (Variable Neighborhood Descent, VND) é um método de busca local que consiste em explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança, aceitando somente soluções de melhora da solução corrente e retornando à primeira estrutura quando uma solução melhor é encontrada (Mladenovic e Hansen, 1997), como descrito no pseudocódigo da Figura 6.

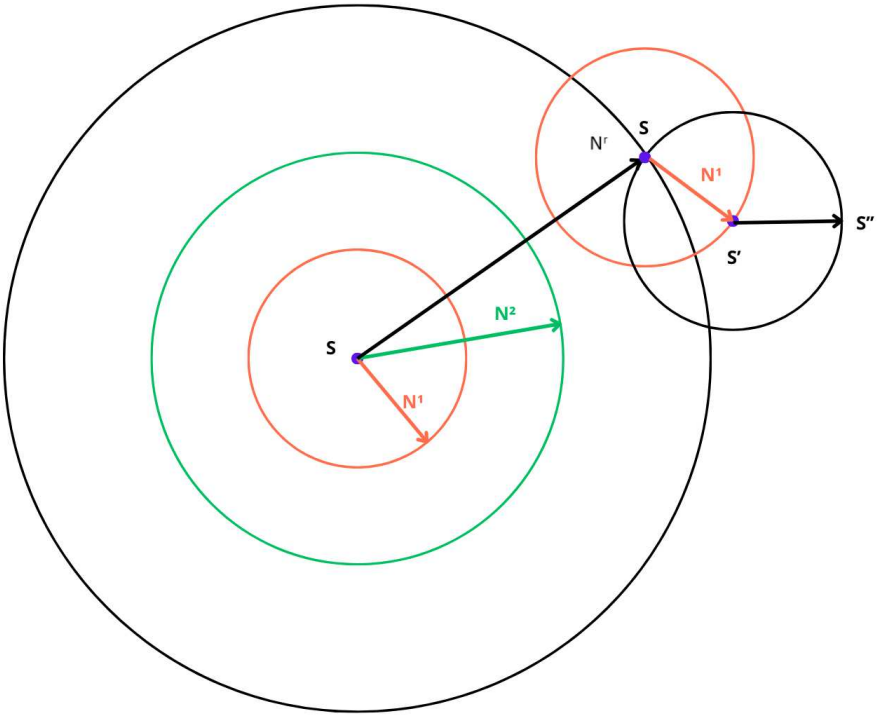
Figura 6 – Pseudocódigo Meta-heurística VND.

Procedimento VND
1. Seja s_0 uma solução inicial e r o número de estruturas de vizinhança
2. $s \leftarrow s_0$ {Solução corrente}
3. $k \leftarrow 1$ {Tipo de estrutura de vizinhança}
4. enquanto $(k \leq r)$ faça
5. Encontre o **melhor vizinho** $s' \in N^{(k)}(s)$
6. se $(f(s') < f(s))$
7. então $s \leftarrow s'$; $k \leftarrow 1$
8. senão $k \leftarrow k + 1$
9. fim-se
10. fim-enquanto
11. Retorne s
Fim VND

Fonte: (Mladenovic e Hansen, 1997)

O método VND percorre estruturas de vizinhança, sucessivamente, até atingir um ótimo local. Primeiro, é preciso definir um conjunto de estruturas de vizinhança $\{N_i \mid 1 \leq i \leq i_{\text{max}}\}$. Seja N_i uma estrutura de vizinhança e s_0 a solução inicial. Se através da vizinhança $N_i(s)$ não for possível atingir um vizinho aprimorante, ou seja, uma solução melhor que a solução atual, a vizinhança N_i é substituída pela vizinhança N_{i+1} (Talbi, 2009).

Figura 7 – Variable Neighborhood Descent



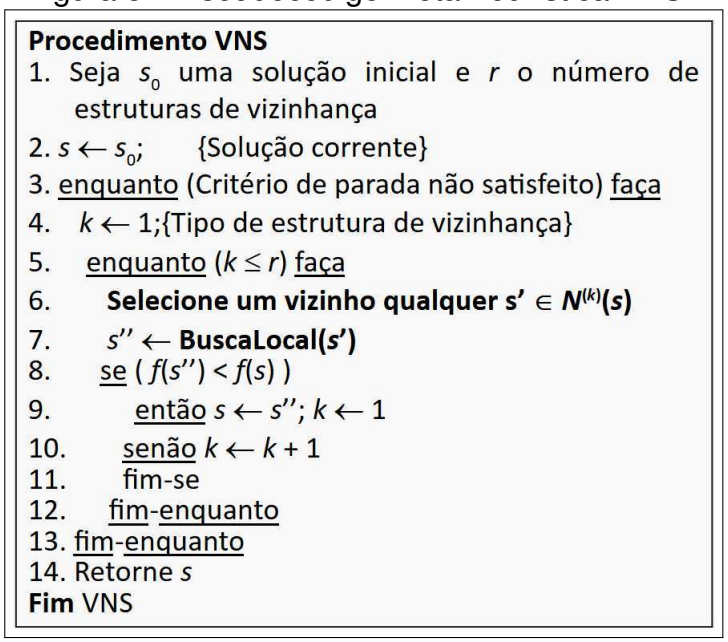
Fonte: Adaptado de Teixeira (2020, p. 23)

Tal procedimento se repete até que um vizinho aprimorante seja encontrado; nesse caso, a heurística retorna para a primeira vizinhança e reinicia a busca, conforme ilustrado na Figura 9. Tal estratégia mostra-se mais eficaz se as vizinhanças utilizadas forem complementares, no sentido de que um ótimo local para uma vizinhança não será um ótimo local para outra vizinhança (Talbi, 2009).

3.3.2 Variable Neighborhood Search (VNS)

O método de Busca em Vizinhança Variável (Variable Neighborhood Search - VNS), proposto por Mladenovic & Hansen (1997), explora o espaço de soluções através da troca sistemática de estruturas de vizinhança. Diferente do VND, o VNS introduz um componente aleatório na geração de vizinhos, uma estratégia que serve como mecanismo de perturbação para escapar de ótimos locais e prevenir a ciclagem, situação que pode ocorrer em métodos puramente determinísticos, como descrito no pseudocódigo da Figura 8.

Figura 8 – Pseudocódigo Meta-heurística VNS.



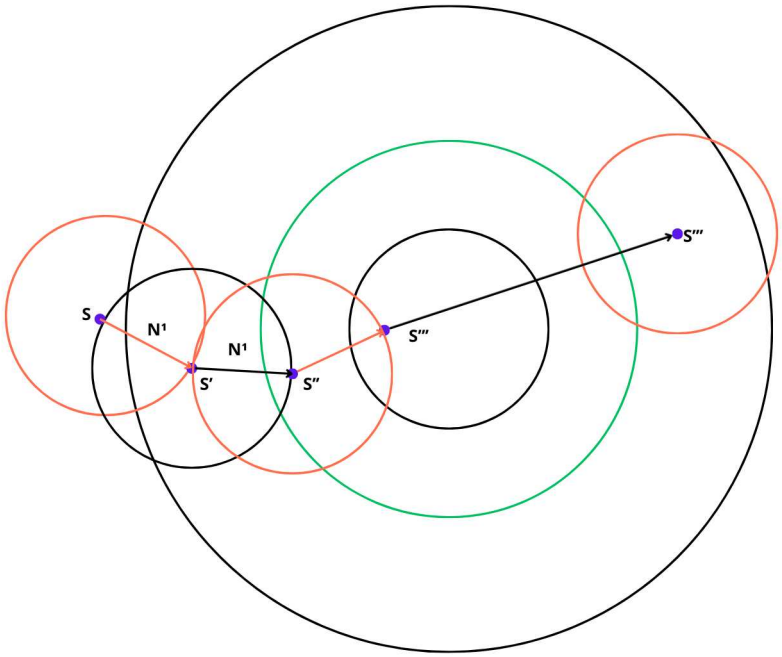
Fonte: (Mladenovic e Hansen, 1997)

O VNS utiliza as mesmas estruturas de vizinhança definidas para o método VND. A cada iteração, o procedimento se inicia com a fase de Perturbação (Shaking), onde a partir da solução corrente s, um vizinho qualquer s' é gerado aleatoriamente dentro de uma estrutura de vizinhança k.

Em seguida, na fase de Busca Local, essa solução perturbada s' é submetida a uma otimização intensiva com o algoritmo VND, resultando em uma nova solução s'', que representa um ótimo local. Por fim, no Critério de Aceitação, a solução s'' é comparada com a solução incumbente s. Se s'' for melhor, a solução incumbente é atualizada para s'' e a busca recomeça da primeira estrutura de vizinhança. Caso contrário, a busca continua a partir de s utilizando a próxima estrutura de vizinhança para a perturbação (Figura 9).

O algoritmo é encerrado quando o número de iterações atinge o limite máximo previamente definido (VNSMax).

Figura 9 – Variable Neighborhood Search



Fonte: Adaptado de Teixeira (2020, p. 23)

A arquitetura do VNS, conforme ilustrada, demonstra como a interação entre a perturbação e a busca local intensiva permite uma exploração robusta do espaço de soluções, sendo eficaz para escapar de ótimos locais (Teixeira, 2020).

Em (Vieira e Taglialenha, 2023) aborda-se o problema de programação de tarefas para robôs móveis autônomos (AMRs) em um ambiente de manufatura flexível. Os robôs são responsáveis por abastecer máquinas e prevenir paradas operacionais por falta de peças. Foi proposto um modelo de Programação Linear Inteira Mista (PLIM) para determinar soluções ótimas em instâncias pequenas, com o objetivo de minimizar o makespan (tempo total de execução). Por se tratar de um problema NP-difícil, foram aplicadas meta-heurísticas baseadas em Variable Neighborhood Search (VNS) e Variable Neighborhood Descent (VND) para resolver instâncias maiores. Os resultados computacionais demonstraram que a abordagem proposta é eficaz e aplicável a cenários mais complexos de manufatura automatizada.

Tendo revisado os conceitos fundamentais das heurísticas construtivas e das meta-heurísticas VND e VNS, o capítulo seguinte apresenta os materiais e métodos, detalhando como esta abordagem teórica foi adaptada e implementada computacionalmente para resolver o problema de roteirização de ordens de serviço do Detrans.

4 MATERIAS E MÉTODOS DE SOLUÇÃO

Neste capítulo apresenta-se o detalhamento dos materiais e métodos considerados no desenvolvimento deste trabalho. Como descrito anteriormente, foram aplicados os métodos exatos e métodos aproximados baseados nas meta-heurísticas Variable Neighborhood Descent e Variable Neighborhood Descent Serch.

4.1 MATERIAIS

Os materiais utilizados neste trabalho consistem nos recursos de hardware, nas fontes de dados e nas ferramentas de software necessárias para a modelagem e experimentação computacional.

- **Hardware:** O desenvolvimento do modelo matemático e os experimentos computacionais foram realizados em um computador pessoal com processador Intel(R) Core(TM) i5-7200U @ 2.50GHz, 8,00 GB de RAM e sistema operacional de 64 bits.
- **Dados do Problema**: Os dados que alimentam o modelo são baseados no contexto da prestação de serviço de pintura para a Prefeitura de Joinville. As informações essenciais, como a lista de Ordens de Serviço (OS), suas localizações (endereços), prioridades e durações estimadas, foram consideradas para a criação dos cenários de teste.
- **Software e Ferramentas Computacionais**: A execução da pesquisa foi suportada por um ecossistema de softwares e ferramentas digitais, cada qual com uma função específica no projeto. A base geográfica do modelo foi construída com o emprego das Interfaces de Programação de Aplicações (APIs) do Google Maps e Open Street Maps. Primeiramente, a API de Geocodificação foi utilizada para converter os endereços das Ordens de Serviço em coordenadas precisas (latitude e longitude). Em seguida, a API de Matriz de Distâncias foi aplicada para calcular os tempos de deslocamento entre todos os pares de locais, considerando as condições reais da malha viária de Joinville. Para a implementação do modelo, o ambiente de desenvolvimento integrado Visual Studio Code (VSCode) forneceu as ferramentas de edição e depuração para os scripts escritos na linguagem Python (versão 3.9). A escolha do Python se deu por sua robustez e vasto suporte de bibliotecas para ciência de dados. As informações primárias das

Ordens de Serviço foram organizadas e tratadas em planilhas do Microsoft Excel, que serviram como fonte de dados para os algoritmos. Finalmente, a redação e formatação deste Trabalho de Conclusão de Curso foram realizadas na plataforma LaTeX, por meio dos editores online Overleaf e Crixet.

Como mencionado na seção de metodologia (3.2), a primeira abordagem metodológica proposta para a solução do problema consiste na aplicação de um método exato. Para isso foi proposta uma formulação matemática de Programação Linear Inteira Mista (PLIM) para resolver instâncias de pequeno porto do problema, a qual será detalhada na seção a seguir.

Na segunda abordagem foi definido um conjunto de instâncias de teste de complexidade crescente. Em todos os cenários, o modelo foi configurado para operar com os parâmetros fixos de 3 equipes e uma jornada de trabalho máxima de 480 minutos. As instâncias variam apenas no número de Ordens de Serviço (OS) a serem atendidas, partindo de um cenário com 5 OS e aumentando progressivamente.

4.1.1 Coleta e Tratamento de Dados

A etapa inicial da implementação envolveu a coleta e o tratamento dos dados operacionais, que serviram como base para a parametrização do modelo. Este processo foi dividido em três fases principais:

- Extração de Dados Primários: A fonte primária de informação consistiu em planilhas de Ordens de Serviço (OS) fornecidas pelo Detrans de Joinville. Destas planilhas, foram extraídos os dados essenciais para cada tarefa, como os endereços, as áreas de pintura (em m^2) e o nível de prioridade.
- Geocodificação e Malha Viária: Para que os endereços pudessem ser usados em cálculos de roteirização, foi necessário convertê-los em coordenadas geográficas (latitude e longitude). Este processo de geocodificação foi realizado utilizando um *script* em Python que consultou as APIs do Google Maps (Geocoding API) e validou os dados com o OpenStreetMap (OSM) para garantir a precisão dos locais.
- Cálculo da Matriz de Custo (Distância): Com as coordenadas do Depósito e de cada OS, o *script* consumiu automaticamente a API Google Maps Distance Matrix. Esta API retorna as distâncias reais de deslocamento entre todos os pares de locais, considerando a malha viária existente, como exemplificado na Tabela 1.

Para fins de exemplificação, os dados de uma instância de cinco OS e três equipes são apresentados a seguir.

Tabela 1 – Matriz de Distâncias (em km) - Instância de 5 tarefas.

Origem	Destino					
	D1	**OS1**	**OS2**	**OS3**	**OS4**	**OS5**
D1	0	12	8	17	11	9
OS1	13	0	7	12	5	13
OS2	9	7	0	14	5	8
OS3	19	13	11	0	11	12
OS4	11	5	3	13	0	9
OS5	9	12	7	11	9	0

Fonte: Elaborado pelo autor (2025), com base nos dados fornecidos pelo Detrans.

Como um dos objetivos do problema consiste em minimizar o tempo (em minutos) de execução das tarefas, a matriz de distâncias foi convertida em uma matriz de tempos de deslocamento em minutos. Este cálculo foi realizado diretamente no *script* do código, considerando a velocidade média das equipes de 40 km/h, utilizando-se a Equação (17), que converte o tempo de horas para minutos:

$$t_{ij}\ (\text{min}) = \left(\frac{d_{ij}\ (\text{km})}{40\ (\text{km/h})} \right) \times 60 \tag{17}$$

Para ilustrar a estrutura final dos dados de entrada, na Tabela 2 apresenta-se a matriz de tempo de deslocamento já calculada para uma instância de teste de pequeno porte, composta pelo depósito (D1) e 5 Ordens de Serviço.

Tabela 2 – Matriz de tempo de deslocamento (minutos) - Instância de 5 tarefas.

Origem	Destino					
	D1	**OS1**	**OS2**	**OS3**	**OS4**	**OS5**
D1	0	18.0	12.0	25.5	16.5	13.5
OS1	19.5	0	10.5	18.0	7.5	19.5
OS2	13.5	10.5	0	21.0	7.5	12.0
OS3	28.5	19.5	16.5	0	16.5	18.0
OS4	16.5	7.5	4.5	19.5	0	13.5
OS5	13.5	18.0	10.5	16.5	13.5	0

Fonte: Elaborado pelo autor (2025).

De forma análoga, o tempo de realização da pintura (tempo de execução da ordem de serviço) de cada tarefa foi calculado a partir de dados primários de área (extraídos das planilhas), acrescido dos tempos fixos de preparação (10 min) e limpeza (10 min). Outros parâmetros, como os prazos de conclusão e as multas por atraso (derivadas da prioridade de cada OS), foram igualmente estruturados em dicionários Python para alimentar o modelo.

4.2 SOLUÇÃO MÉTODO EXATO - MODELAGEM PLIM

Nesta seção propõe-se a formulação matemática de um modelo em três estágios para a resolução do problema de designação, sequenciamento (scheduling) e

roteirização de ordens de serviço para equipes de pintura. O objetivo é, primeiramente, alocar as tarefas de forma a balancear a carga de trabalho entre as equipes e, em seguida, otimizar a rota de cada equipe para minimizar o tempo de deslocamento.

Diante deste cenário, foi desenvolvido um modelo de Programação Linear Inteira Mista (PLIM), denominado Problema Integrado de Designação, Roteirização e Escalonamento de Serviços (PIDRES), para tratar instâncias de pequeno porte. O modelo proposto foi projetado para determinar a designação, a programação e a roteirização das rotas para realização dos serviços de pintura de sinalização horizontal em vias urbanas no município de Joinville, buscando minimizar uma função objetivo que pondera o custo de deslocamento entre as tarefas, o valor (bônus) gerado pela execução das ordens de serviço e a penalidade por atrasos em relação aos prazos.

A seguir descreve-se os parâmetros e váriáveis utilizados na elaboração do modelo dado em (18) a (30).

Conjuntos:

- V: Conjunto de todos os nós, $V = \{0\} \cup F$, onde o nó 0 representa o Depósito;
- F: Conjunto de nós que representam as tarefas (O.S.), $F = \{1, 2, ..., n\}$;
- K: Conjunto de equipes, $K = \{1, 2, ..., m\}$.

Índices:

- i, j: Índices para os nós, onde $i, j \in V$;
- k: Índice para as equipes, onde $k \in K$.

Parâmetros:

- tv_{ij}: Tempo de viagem do nó i para o nó j;
- tp_i: Tempo processamento (realização) da tarefa i;
- P_i: Valor binário que indica se a tarefa i é prioritária ($P_i = 1$) ou não ($P_i = 0$);
- d_i: Prazo final, para a conclusão da tarefa i;
- T_k: Tempo máximo de jornada de trabalho disponível para a equipe k;
- N: Número total de tarefas, $N = |F|$;
- M: Constante positiva grande (Big M);
- α: Fator de custo para o tempo de viagem;
- β_{base}: Bônus base por realizar qualquer tarefa;
- β_{prio}: Bônus adicional por realizar uma tarefa prioritária;
- γ: Fator de penalidade por minuto de atraso.

Variáveis de Decisão

- $y_{ijk} \in \{0, 1\}$: Variável binária que assume valor 1 se a equipe k viaja diretamente do nó i para o nó j, e 0 caso contrário;
- $z_{ik} \in \{0, 1\}$: Variável binária que assume valor 1 se a tarefa i é atribuída à equipe k, e 0 caso contrário;
- $x_{ik} \geq 0$: Variável que representa o tempo de início do serviço na tarefa i pela equipe k;

- $u_i \in \mathbb{R}$: Variável auxiliar para a eliminação de sub-rotas;
- $s_{ik}^{+} \geq 0$: Variável que representa os minutos de atraso da tarefa i atribuída à equipe k;
- $s_{ik}^{-} \geq 0$: Variável que representa os minutos de adiantamento da tarefa i atribuída à equipe k.

Função Objetivo:

O objetivo do modelo é minimizar o custo total Z, composto pelo custo de viagem, bônus por tarefas e penalidade por atrasos. O bônus é subtraído (tornado negativo) para que o modelo seja incentivado a maximizá-lo. Este bônus só é concedido caso a tarefa seja de fato realizada, o que é controlado pela multiplicação pela variável de decisão binária z_{ik}.

$$\text{Minimizar} \quad Z = \alpha \sum_{k \in K} \sum_{i \in F} \sum_{j \in V} tv_{ij} \cdot y_{ijk} - \sum_{k \in K} \sum_{i \in F} (\beta_{base} + (\beta_{prio} \cdot P_i)) \cdot z_{ik} + \gamma \sum_{k \in K} \sum_{i \in F} s_{ik}^{+} \quad (18)$$

Restrições:

Propõe-se as seguintes restrições para o modelo:

Jornada Máxima por Equipe:

O tempo total de trabalho de cada equipe não pode exceder sua jornada máxima.

$$\sum_{i \in V} \sum_{j \in V} tv_{ij} \cdot y_{ijk} + \sum_{i \in F} tp_i \cdot z_{ik} \leq T_k, \quad \forall k \in K \quad (19)$$

Atribuição e Cobertura:

Cada tarefa deve ser atribuída a exatamente uma equipe.

$$\sum_{k \in K} z_{ik} \leq 1, \quad \forall i \in F \quad (20)$$

Conservação de Fluxo:

Se uma tarefa é atribuída a uma equipe, uma rota deve chegar e sair daquela tarefa.

$$\sum_{j \in V} y_{jik} = z_{ik}, \quad \forall i \in F, \forall k \in K \quad (21)$$

$$\sum_{i \in V} y_{ijk} = z_{ik}, \quad \forall j \in F, \forall k \in K \quad (22)$$

Regras do Depósito:

Cada equipe pode sair do depósito no máximo uma vez, e se sair, deve retornar.

$$\sum_{j \in F} y_{0jk} \leq 1, \quad \forall k \in K \quad (23)$$

$$\sum_{j \in F} y_{0jk} = \sum_{i \in F} y_{i0k}, \quad \forall k \in K \quad (24)$$

Sequenciamento Temporal:
O tempo de início de uma tarefa só pode iniciar depois da conclusão da tarefa anterior, acrescentado do tempo de viagem até a tarefa atual.

$$x_{jk} \geq x_{ik} + tp_i + tv_{ij} - M(1 - y_{ijk}), \quad \forall i, j \in F, i \neq j, \forall k \in K \tag{25}$$

$$x_{ik} \leq x_{jk} + tp_j + tv_{ji} + M \cdot (1 - y_{jik}), \quad \forall i, j \in V, i \neq j, \forall k \in K \tag{26}$$

Vínculo entre Tempo e Atribuição:
O tempo de início só pode ser maior que zero se a tarefa for atribuída.

$$x_{ik} \leq M \cdot z_{ik} \quad \forall i \in F, \forall k \in K \tag{27}$$

Cumprimento do Prazo de Execução:
Mede o atraso ou adiantamento em relação ao prazo.

$$x_{ik} + tp_i - s_{ik}^{+} + s_{ik}^{-} \leq d_i + M(1 - z_{ik}), \quad \forall i \in F, \forall k \in K \tag{28}$$

Eliminação de Sub-rotas:
Proíbe que uma rota inicie e finalize no mesmo nó.

$$y_{iik} = 0, \quad \forall i \in V, \forall k \in K \tag{29}$$

Eliminação de Sub-rotas:
Ela proíbe que a mesma equipe crie duas ou mais rotas separadas que não se conectam.

$$u_i - u_j + N \cdot y_{ijk} \leq N - 1, \quad \forall i, j \in F, i \neq j, \forall k \in K \tag{30}$$

4.2.1 Implementação Computacional do Modelo PLIM

A abordagem de solução exata foi materializada pela implementação computacional do modelo PLIM (18)-(30).

A implementação foi desenvolvida em Python, utilizando a biblioteca `gurobipy` (versão 11.0) como interface para o *solver* Gurobi Optimizer.

Com os parâmetros devidamente estruturados na etapa anterior, deu-se a tradução da formulação matemática para a sintaxe da biblioteca `gurobipy`.

As variáveis de decisão do modelo foram criadas utilizando a função `addVars`, que permite a definição do domínio, do tipo e do nome de cada variável, conforme ilustrado no Código 4.1.

Código 4.1 – Exemplo de declaração de variável no Gurobi.

```
# Declaracao da variavel de rota y[i,j,k]
y = model.addVars(NOS, NOS, 0, vtype=GRB.BINARY, name="Y")
```

A Função Objetivo definida na Equação 18 (combinando custo de viagem, bônus e penalidade), foi declarada no modelo através do método `setObjective`. O Código 4.2 ilustra a sua implementação.

Código 4.2 – Implementação da Função Objetivo (Equação 18).

```
# Implementacao dos tres componentes da FO
custo_viagem = 0.6 * gp.quicksum(T[i,j] * y[i,j,k] for i in NOS for
    j in NOS for k in 0)
bonus_tarefas = gp.quicksum((40 + 20 * PRIORIDADE[i]) * z[i,k] for
   i in TAREFAS for k in 0)
custo_multa = 0.4 * gp.quicksum(s_plus[j,k] for j in TAREFAS for k
   in 0)

# Definicao da FO
model.setObjective(custo_viagem - bonus_tarefas + custo_multa, GRB.
   MINIMIZE)
```

Cada restrição do modelo (19)-(30) foi implementada em Python para utilizar o solver Gurobi utilizando o método `addConstr`. Como exemplo, a restrição híbrida de sequenciamento temporal (Código 4.3), que é central para a lógica do modelo, integra o tempo de serviço e o tempo de deslocamento.

Código 4.3 – Exemplo de implementação da restrição de sequenciamento temporal.

```
# Restricao de sequenciamento temporal (Big-M)
for i in nos:
    for j in tarefas:
        if i != j:
            for k in equipes:
                model.addConstr(
                    x[j, k] >= x[i, k] + tempo_servico[i] +
                       tempo_deslocamento[i, j] - M * (1 - y[i, j,
                       k])
                )
```

A implementação completa do modelo proposto, incluindo a estruturação de todos os dados e a formatação da saída dos resultados, pode ser encontrada no Apêndice A.

4.2.2 Análise de complexidade e limitações do Modelo PLI

A complexidade computacional do problema em estudo é compreendida a partir de sua decomposição em três problemas clássicos da Pesquisa Operacional. Primeiramente, ele incorpora as características do Problema de Designação Generalizado, que consiste em atribuir um conjunto de tarefas a agentes com capacidades limitadas, sendo este um problema NP-Difícil.

Adicionalmente, o problema se enquadra na classe de *Scheduling* em Máquinas Paralelas com tempos de *setup* dependentes da sequência, onde o deslocamento entre tarefas é modelado como *setup*, o que eleva exponencialmente a complexidade, conforme a classificação de Pinedo (2016). Além disso, o modelo incorpora o Problema de Roteirização de Veículos (VRP). A integração destas três classes de problemas NP-Difíceis comprova que o problema geral é intratável por métodos exatos para instâncias de grande porte, justificando a necessidade de abordagens heurísticas.

4.3 SOLUÇÃO MÉTODOS APROXIMADOS: VND E VNS

Como mencionado anteriomente, o PIDRES pode ser decomposto em problemas clássicos da literatura: o Problema de Designação (Assignment Problem) (Arenales et al., 2007), o Problema de Escalonamento em Máquinas Paralelas (Parallel Machine Scheduling) (Pinedo, 2016) e o Problema de Roteirização de Veículos (PRV) (Toth e Vigo, 2014), e portanto, pertence à classe de problemas NP-difícil, o que inviabiliza o uso de métodos exatos para encontrar soluções ótimas em instâncias de grande porte dentro de um tempo computacional aceitável (Toth e Vigo, 2014).

Diante disso foram desenvolvidos algoritmos aproximados nas meta-heurísticas Busca em Vizinhança Variável (VNS - Variable Neighborhood Search) e Busca em Vizinhança Variável de Descida (VNS - Variable Neighborhood Descend). Segundo (Mladenovic e Hansen, 1997), uma técnica robusta e amplamente aplicada a problemas de roteirização. A principal força da VNS reside na sua capacidade de escapar de ótimos locais por meio da exploração sistemática de múltiplas estruturas de vizinhança (Taglialenha e Lázaro, 2019).

A seção a seguir apresenta-se os detalhes da implementação computacional desta abordagem.

4.3.1 Solução Inicial e Estrutura de Dados

Para que as meta-heurísticas de melhoria (VND e VNS) possam operar, é necessário primeiro gerar uma solução inicial que seja factível e, preferencialmente, de boa qualidade. Em vez de partir de uma solução puramente aleatória, que poderia estar em uma região muito ruim do espaço de busca, foi implementada uma Heurística Construtiva de Inserção Gulosa.

A heurística opera de forma iterativa, começando com as rotas de todas as equipes vazias e uma lista de todas as Ordens de Serviço (OS) a serem alocadas. A cada passo, o algoritmo toma uma decisão gulosa, ou seja, escolhe a ação que parece ser a melhor naquele momento, até que todas as tarefas sejam alocadas.

No Código 4.4 tem-se o pseudo-código que sintetiza essa heurística, ilustrando-se o trecho central desta heurística, o laço principal (while) e a busca aninhada (for) que implementam a lógica de decisão para encontrar a inserção de menor custo marginal a cada passo.

Código 4.4 – Heurística Construtiva de Inserção Gulosa.

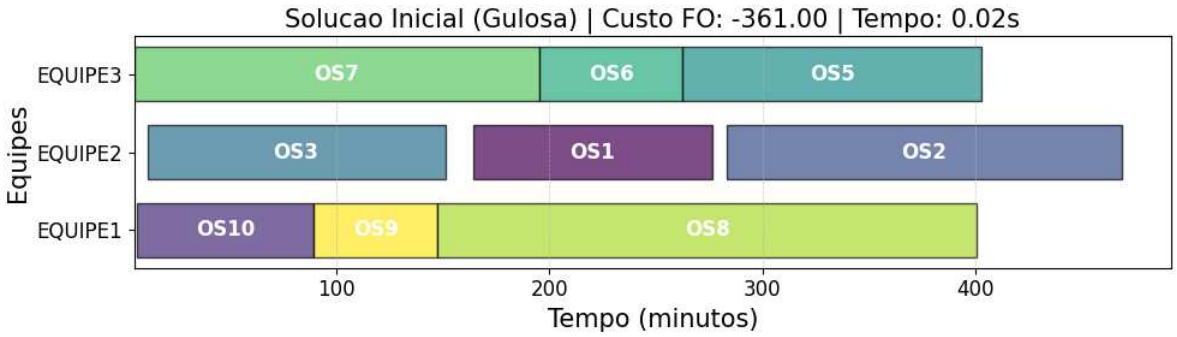
```
# Inicia com rotas vazias e todas as tarefas pendentes
solucao_atual = {equipe: [] for equipe in O}
tarefas_restantes = copy.deepcopy(TAREFAS)
# Loop principal: continua enquanto houver tarefas para alocar
while tarefas_restantes:
    melhor_custo_delta, melhor_posicao, tarefa_a_mover = float('inf
        '), None, None

 # Busca exaustiva pela melhor insercao possivel
    for tarefa in tarefas_restantes:
        for equipe in O:
            for i in range(len(solucao_atual[equipe]) + 1):
                custo_antes, _, = calcular_custo_e_validade(
                    solucao_atual)

                # Simula a insercao em uma copia da solucao
                solucao_teste = copy.deepcopy(solucao_atual)
                solucao_teste[equipe].insert(i, tarefa)
                custo_depois, validade, = calcular_custo_e_validade
                    (solucao_teste)

                # Verifica se a insercao eh valida e se eh a melhor
                    encontrada ate agora
                if validade and (custo_depois - custo_antes) <
                    melhor_custo_delta:
                    melhor_custo_delta = custo_depois - custo_antes
                    melhor_posicao = (equipe, i)
                    tarefa_a_mover = tarefa

    # Efetiva o melhor movimento encontrado na iteracao
    if melhor_posicao:
```

```
            solucao_atual[melhor_posicao[0]].insert(melhor_posicao[1],
                tarefa_a_mover)
            tarefas_restantes.remove(tarefa_a_mover)
        else:
            # Se nenhuma insercao for possivel, encerra (evita loop
                infinito)
            break
```

Ao final da busca exaustiva, a melhor inserção encontrada é efetivada: a tarefa é permanentemente adicionada à rota da equipe escolhida e removida da lista de tarefas pendentes. O processo então se repete, buscando a melhor inserção para as tarefas restantes.

O procedimento termina quando todas as tarefas são designadas, compondo então uma solução factível para o problema. Na Figura 10 tem-se a representação de uma solução inicial para a instância I3 (10 ordens e três equipes).

Figura 10 – Solução Inicial



Fonte: Autor (2025)

Ao final deste procedimento, uma solução completa e factvel é obtida, servindo como um excelente ponto de partida para ser refinada pelas fases de busca local (VND) e perturbação (*Shake*) da meta-heurística VNS.

4.3.2 Avaliação da Função Objetivo (FO)

A Função Objetivo (FO) é o componente central do modelo de otimização, pois é responsável por traduzir os objetivos estratégicos e operacionais do problema em uma única métrica quantitativa. É através da avaliação desta função que o algoritmo consegue comparar diferentes soluções, ou seja, diferentes conjuntos de rotas, e determinar qual delas é melhor.

A função considerada no algoritmo implementado, recebe uma solução candidata como entrada e retorna seu valor numérico. O objetivo do modelo é minimizar este valor. A estrutura da FO foi projetada para balancear três componentes, muitas vezes conflitantes, que refletem as prioridades do problema: o custo operacional, o valor gerado pela execução das tarefas e a qualidade do serviço em relação aos prazos.

A expressão matemática da FO, apresentada na Equação 18, é composta por três termos principais, cada um com um peso que define sua importância relativa.

$$\text{Minimizar} \quad Z = (\alpha \cdot \text{Custo Viagem}) - (\text{Bônus Tarefas}) + (\gamma \cdot \text{Penalidade Atraso}) \quad (23)$$

1. **Custo de Deslocamento ($\alpha \cdot \sum t_{ij} \cdot y_{ijk}$):** Este termo representa o tempo de deslocamento e o tempo gasto na execução do trabalho. Ele é calculado somando-se o tempo de viagem de todos os arcos percorridos nas rotas da solução e o tempo de execução das OS. O parâmetro α (definido como 0.6 no modelo) funciona como um fator de ponderação para converter o tempo em um custo. O objetivo é minimizar este componente.
2. **Bônus por Tarefas Realizadas ($\sum(\beta_{\text{base}} + (\beta_{\text{prio}} \cdot P_i)) \cdot z_{ik}$):** Este componente representa o valor gerado ao se completar as Ordens de Serviço. O modelo é incentivado a realizar o maior número de tarefas possível, especialmente aquelas de alta prioridade. Para alcançar a maximização deste valor dentro de uma função de minimização, este termo é subtraído do custo total. Este componente representa o valor gerado ao se completar as Ordens de Serviço. O modelo é incentivado a realizar o maior número de tarefas possível, especialmente aquelas de alta prioridade. Para alcançar a maximização deste valor dentro de uma função de minimização, este termo é subtraído do custo total. O bônus é composto por:
   - Um valor base ($\beta_{\text{base}} = 40$) por qualquer tarefa concluída.
   - Um bônus adicional ($\beta_{\text{prio}} = 20$) para tarefas marcadas como prioritárias ($P_i = 1$).
3. **Penalidade por Atraso ($\gamma \cdot \sum s_{ik}^{+}$):** Este termo introduz o conceito de conformidade com os prazos acordados. Ele penaliza a solução por atrasos (s_{ik}^{+}) na conclusão de uma tarefa em relação ao seu prazo final (d_i). O parâmetro γ (definido como 0.4) quantifica o quão custoso é um atraso. Este componente força o modelo a não apenas executar as tarefas, mas a fazê-lo dentro do tempo esperado, sempre que possível.

4.3.2.1 Justificativa da Calibração dos Pesos

A eficácia do modelo depende de direcionar ao algoritmo quais são as prioridades estratégicas do Detrans. Isso é feito pela calibração dos pesos (α, β, γ), que definem as regras para as escolhas que o algoritmo fará. A lógica da calibração segue três prioridades principais:

Regra 1: Eficiência no Uso de Recursos Públicos ($\alpha > \gamma$)

A principal diretriz é a priorização da eficiência operacional sobre a pontualidade. Os pesos foram definidos como Custo da Operação ($\alpha = 0.6$) e Multa por

Atraso ($\gamma = 0.4$). Ao estabelecer $\alpha > \gamma$, o modelo é instruído a priorizar o melhor uso dos recursos da equipe (tempo de trabalho, deslocamento). Esta decisão é pautada pelo Princípio da Eficiência, que rege a administração pública brasileira, conforme estabelecido no Art. 37 da Constituição Federal 1988. Na prática, o algoritmo entende que é preferível incorrer em um atraso pontual em um serviço não emergencial (custo 0.4) se isso gerar uma economia de tempo de rota (custo 0.6). Essa economia, por sua vez, maximiza a capacidade da equipe para executar mais Ordens de Serviço com os mesmos recursos públicos.

Prioridade 2: Foco em Demandas Urgentes (β_{prio})

A segunda diretriz é garantir que demandas urgentes sejam atendidas. O bônus por realizar uma OS foi dividido em um valor base ($\beta_{\text{base}} = 40$) e um bônus adicional por prioridade ($\beta_{\text{prio}} = 20$).Com essa calibração, uma OS prioritária (valendo 60 pontos no total) torna-se levemente mais importante que uma OS normal (40 pontos). Isso força o algoritmo a se esforçar para encontrar formas de incluir as OS prioritárias nas rotas.

Prioridade 3: Limite de Esforço Justificado (β/α)

Por fim, o modelo deve decidir se uma OS vale a pena ser incluída na rota. O algoritmo faz isso comparando o bônus que recebe por fazer a tarefa (Prioridade 2) com o custo que gasta para executá-la (Prioridade 1).Isso cria um ponto de equilíbrio. O modelo só incluirá uma OS se o benefício de fazê-la for maior que o seu custo. Dessa forma, o algoritmo rejeita automaticamente Ordens de Serviço que exigem um esforço desproporcional (como um grande desvio de rota para uma OS de baixo valor), garantindo que o tempo da equipe seja sempre usado de forma economicamente justificada.

Conclusão da Calibração

Em resumo, a calibração ($\alpha = 0.6, \gamma = 0.4, \beta_{\text{base}} = 40, \beta_{\text{prio}} = 20$) traduz as regras em um sistema de pontos que força o algoritmo a: (1) priorizar a eficiência de recursos ($\alpha > \gamma$); (2) priorizar demandas de maior valor (β_{prio}); e (3) trabalhar dentro de um limite de esforço justificado.

4.3.2.2 Implementação Computacional

No Código dos métodos desenvolvidos, a Função Objetivo (FO) simula a execução das rotas de uma determinada solução, passo a passo, para calcular de forma precisa cada um dos componentes do custo.

Uma espécie de relógio é mantido para cada equipe, rastreando o tempo de viagem e de serviço acumulados. A cada tarefa visitada, o custo de deslocamento (α) é somado ao custo total; o bônus (β) correspondente àquela tarefa é subtraído; e o tempo de conclusão é comparado ao prazo (d_i) para calcular o atraso e somar a penalidade (γ) correspondente. Simultaneamente, a função verifica a viabilidade da solução, garantindo que a duração total não exceda a jornada máxima (T_k).

O Código 4.5 ilustra como esses três componentes são calculados e combinados dentro da função `calcular_custo_e_validade` para obter o custo final da solução.

Código 4.5 – Implementação da FO na heurística.

```
# --- Dentro do loop de simulacao da rota (para cada tarefa) ---

# 1. Custo de Viagem (alpha = 0.6)
custo_viagem_obj += 0.6 * tempo_de_viagem_trecho

# 2. Bonus por Tarefa (beta_base=40, beta_prio=20)
bonus_tarefas_obj += (40 + 20 * PRIORIDADE[tarefa_atual])

# 3. Penalidade por Atraso (gamma = 0.4)
atraso = max(0, tempo_conclusao - d_i[tarefa_atual])
custo_multa_atraso_obj += 0.4 * atraso

# --- Apos o loop de todas as equipes ---

# Combina os 3 componentes do PLIM
custo_total_obj += (custo_viagem_obj - bonus_tarefas_obj +
   custo_multa_atraso_obj)

return custo_total_obj
```

Dessa forma, a função `calcular_custo_e_validade` é a tradução fiel da expressão matemática do PLIM (Equação 23). Ela permite que a heurística VNS/VND avalie milhares de soluções vizinhas por segundo, sempre guiada pelo mesmo critério de otimalidade definido no modelo matemático.

4.3.3 Busca Local e Vizinhanças

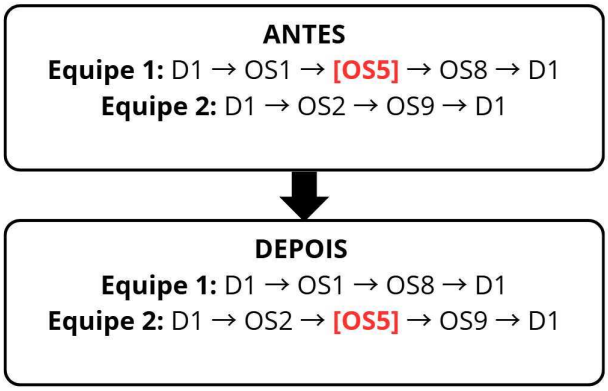
Após a obtenção da solução inicial factível, aplica-se buscas locais de descida em sua vizinhança. E para escapar de ótimos locais, que exploram o espaço de soluções de forma inteligente. A implementação é composta por três elementos centrais: as

estruturas de vizinhança (os "movimentos"), o motor de busca local (VND) e o algoritmo principal que gerencia a busca (VNS).

A busca por soluções melhores é realizada através da exploração sistemática de vizinhanças, que são conjuntos de soluções similares à atual, alcançadas por meio de pequenas modificações chamadas de movimentos ou operadores. Foram implementados três operadores principais, cada um definindo uma forma diferente de explorar o espaço de soluções:

- Realocação (Relocate): Move uma tarefa para uma nova posição, seja dentro da mesma rota (intra-rota) ou para a rota de outra equipe (inter-rotas).

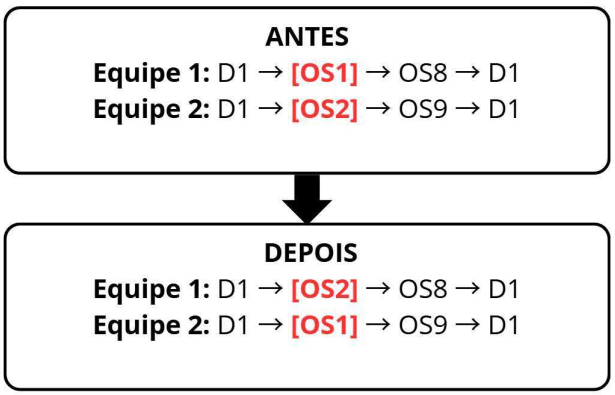
Figura 11 – Movimento de Realocação



Fonte: Autor (2025)

- Troca (Exchange): Permuta duas tarefas entre as rotas de duas equipes distintas (inter-rotas).

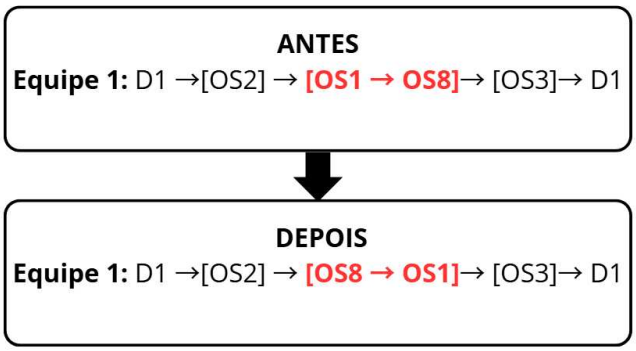
Figura 12 – Movimento de Troca



Fonte: Autor (2025)

- 2-Opt: Inverte um segmento de uma rota para otimizar o percurso, uma técnica clássica para problemas de roteirização.

Figura 13 – Movimento 2-Opt



Fonte: Autor (2025)

O código 4.6 ilustra a lógica conceitual de um operador de vizinhança. O processo consiste em criar uma cópia da solução atual, aplicar a modificação e, em seguida, chamar a função de avaliação para verificar se a "solução vizinha"é válida e se representa uma melhoria.

Código 4.6 – Lógica central de um operador de vizinhança (Relocate)

```
# A funcao recebe a solucao atual para tentar melhora-la
def relocate(solucao):
    melhor_custo, _ = calcular_custo_e_validade(solucao)

    # Lacos 'for' para testar todas as combinacoes de movimento
    for equipe1, rota1 in solucao.items():
        for i in range(len(rota1)):
            # ... lacos para equipe2 e posicao j ...

            # Cria uma copia para nao alterar a solucao original
            solucao_vizinha = copy.deepcopy(solucao)

            # Aplica o movimento de realocacao
            tarefa_removida = solucao_vizinha[equipe1].pop(i)
            solucao_vizinha[equipe2].insert(j, tarefa_removida)

            # Avalia a nova solucao
            custo_vizinho, validade_vizinho =
               calcular_custo_e_validade(solucao_vizinha)

            # Se for valida e melhor, retorna imediatamente (
               estrategia First Improvement)
            if validade_vizinho and custo_vizinho < melhor_custo:
                return solucao_vizinha, custo_vizinho
```

```
    # Se nenhum movimento melhorou a solucao, retorna a original
    return solucao, melhor_custo
```

A exploração sistemática das vizinhanças é orquestrada pelo procedimento de Descida em Vizinhança Variável (VND). O VND atua como um motor de busca local intensiva, cujo objetivo é encontrar o ótimo local de uma determinada solução, ou seja, um ponto a partir do qual nenhum dos movimentos implementados consegue gerar uma melhoria.

Conforme implementado no Código, o algoritmo opera sobre uma lista pré-definida de operadores de vizinhança. Ele aplica o primeiro operador (*relocate*) até não encontrar mais melhorias. Em seguida, passa para o segundo (*exchange*), e assim por diante. Se, em qualquer ponto, uma melhoria é encontrada, o processo retorna imediatamente ao primeiro operador da lista para reiniciar a busca a partir da nova solução aprimorada. O procedimento termina quando uma passagem completa por todos os operadores não gera nenhuma melhoria.

Para evitar que a busca fique presa em ótimos locais de baixa qualidade, o VND é incorporado a uma meta-heurística mais ampla, a Busca em Vizinhança Variável (VNS). O VNS equilibra a busca intensiva do VND com a diversificação, que é a capacidade de saltar para outras áreas do espaço de soluções.

O ciclo principal do VNS, ilustrado no Código 4.7, alterna entre duas fases:

- Perturbação: A melhor solução encontrada até o momento é intencionalmente "chacoalhada"através da aplicação de um número *k* de movimentos aleatórios. O objetivo é deslocar a solução para uma nova região do espaço de busca.
- Busca Local: O procedimento VND é aplicado a esta nova solução perturbada para encontrar o ótimo local daquela região.

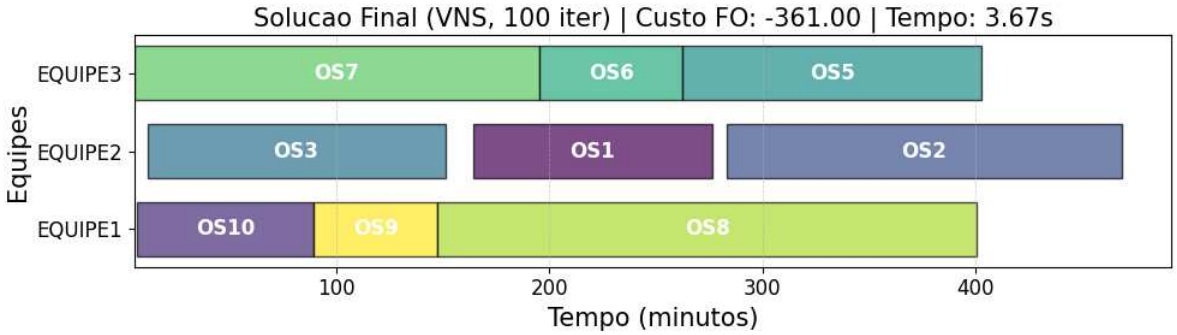
Código 4.7 – Lógica central do algoritmo VNS

```
# Loop principal do VNS
for i in range(max_iteracoes):
    k = 1
    while k <= k_max:
        # 1. Perturba a melhor solucao encontrada ate agora
        solucao_agitada = shake(melhor_solucao_global, k)

        # 2. Aplica a busca local intensiva (VND)
        nova_solucao_local = vnd(solucao_agitada)
        novo_custo_local, validade = calcular_custo_e_validade(
            nova_solucao_local)

        # 3. Compara e, se for o caso, atualiza a melhor solucao
        if validade and novo_custo_local < melhor_custo_global:
```

```
            melhor_solucao_global = nova_solucao_local
            melhor_custo_global = novo_custo_local
            k = 1 # Se melhorou, volta para a vizinhanca k=1
        else:
            k += 1 # Se nao melhorou, tenta uma perturbacao mais
                forte
```

O novo ótimo local encontrado é então comparado com a melhor solução global. Se houver uma melhora, a melhor solução global é atualizada e o processo de perturbação é reiniciado com uma força mínima (k=1). Caso contrário, a força da perturbação é aumentada (k=k+1) para tentar escapar da bacia de atração atual. Este ciclo é repetido por um número pré-definido de iterações.

Figura 14 – Solução com aplicação de VND e VNS



Fonte: Autor (2025)

5 Resultados e Discussão

Neste capítulo são apresentados e discutidos os resultados obtidos com a aplicação do método exato, o modelo de Programação Linear Inteira Mista (PLIM), e das meta-heurísticas VND e VNS.

Apresenta-se uma análise comparativa de desempenho, avaliando não apenas a qualidade da solução e o custo computacional, mas também as características intrínsecas de cada abordagem frente a instâncias de complexidade crescente.

5.1 RESULTADOS OBTIDOS COM O MODELO PLIM

A utilização do modelo PLIM teve o propósito fundamental de estabelecer um padrão de otimalidade, ou seja, um padrão de referência contra o qual a eficácia de qualquer método aproximado pode ser rigorosamente medida.

5.1.1 Análise de Desempenho e Limites do PLIM

Na Tabela 3 são ilustrados os resultados do modelo PLIM (18)-(30) para instâncias de pequeno porte.

Tabela 3 – Resultados do método exato para instâncias de pequeno porte.

Instância	Nº de Tarefas	Valor da F.O.	Tempo Computacional (s)
I1	5	-175.0	0.4
I2	8	-280.4	2.4
I3	10	-361.0	4.2
I4	12	-376.2	28.7
I5	15	Tamanho Excedido	Tamanho Excedido

A análise da Tabela 3 deixa explícito o dilema clássico de métodos exatos. O salto de 4.2 segundos (I3) para 28.7 segundos (I4), com um acréscimo de apenas duas tarefas, já sinaliza o crescimento não-polinomial do tempo. O ponto de inflexão ocorre na instância I5 (15 tarefas), onde a explosão combinatória torna o problema intratável. Este resultado prático não é uma falha, mas corrobora a natureza NP-difícil do problema, validando a necessidade de se recorrer a métodos heurísticos para instâncias de porte realista.

5.1.2 Análise da solução para validação do modelo (Instância I2)

Tendo estabelecido os limites computacionais, é importante verificar se a solução obtida faz sentido na prática, para considerar o modelo proposto (5)-(22) validado. Para este fim, a Instância I2 (com 8 tarefas e 3 equipes) foi selecionada como o principal padrão de comparaç

6 Conclusão

Neste trabalho apresentaram-se os resultados do estudo do Problema Integrado de Designação, Roteirização e Escalonamento de Serviços (PIDRES), focado no processo de pintura de sinalização horizontal do Detran de Joinville. O problema consistiu em definir quais ordens de serviço deveriam ser designadas para cada equipe, bem como a sequência (escalonamento) e a rota (roteirização) de realização dessas tarefas. O objetivo foi minimizar uma função objetivo ponderada que considera o custo de deslocamento, o bônus pela execução de tarefas (prioritárias ou não) e a penalidade por atrasos, atendendo restrições operacionais como a jornada máxima de trabalho, sendo classificado como um problema combinatório NP-Difícil.

Apresentou-se um modelo de Programação Linear Inteira Mista (PLIM) que determinou a solução ótima global para instâncias de pequeno porte (até 12 tarefas). Os testes (Tabela 3) confirmaram a natureza NP-Difícil do PIDRES, visto que instâncias com 15 tarefas não puderam ser resolvidas pelo método exato em tempo hábil, justificando a necessidade de abordagens aproximadas.

Para tratar instâncias maiores, aplicou-se uma Heurística Construtiva Gulosa, capaz de determinar rapidamente soluções iniciais viáveis em baixo tempo computacional, que serviram como ponto de partida para as meta-heurísticas. Também se propuseram algoritmos meta-heurísticos baseados em busca local em vizinhanças variáveis (*Variable Neighborhood Search* – VNS, e *Variable Neighborhood Descent* – VND), tendo como solução inicial as soluções obtidas pela heurística gulosa. A análise comparativa demonstrou que o VNS apresentou desempenho superior ao VND. Verificou-se que o VND (determinístico) frequentemente convergiu para ótimos locais de baixa qualidade (custo de -239.87 na Instância I2). O VNS, através do seu mecanismo de perturbação, foi capaz de escapar desses pontos e encontrar soluções melhores (-245.27 na I2).

Quanto ao esforço computacional do VNS, a análise do número de repetições indicou que cerca de 100 repetições representam um ponto de equilíbrio pragmático entre a qualidade da solução e o tempo de execução, obtendo a maior parte das melhorias sem o custo excessivo de 1000 repetições.

As principais contribuições deste trabalho foram:

- A formulação de um modelo matemático (PLIM) para o Problema Integrado de Designação, Roteirização e Escalonamento de Serviços (PIDRES), adaptado à

realidade do Detran;

- A implementação de uma Heurística Construtiva Gulosa para a criação rápida de soluções iniciais viáveis;
- A implementação e análise comparativa das meta-heurísticas VND e VNS, demonstrando a superioridade do VNS em escapar de ótimos locais para este problema;
- Uma análise de *trade-off* (qualidade vs. tempo) para a parametrização do VNS (número de repetições), validando sua eficiência para uso prático.

Como principal limitação do estudo, destaca-se a impossibilidade de realizar uma comparação quantitativa direta com o processo atual do Detran, devido à falta de dados históricos estruturados (como horários e sequências de execução). Desta forma, a validação da eficácia dos métodos aproximados foi realizada contra o padrão da solução ótima do PLIM.

Adicionalmente, destaca-se que os pesos utilizados na função objetivo foram definidos a partir de uma calibração empírica, buscando alinhar o desempenho nos testes computacionais às necessidades operacionais e prioridades reportadas pelo Detran. Embora essa abordagem tenha sido funcional para representar a realidade do problema, ela gera certa subjetividade. Existem métodos mais robustos e reconhecidos cientificamente que podem ser empregados para determinar pesos de maneira sistemática, como o Processo Analítico Hierárquico (AHP), o método de entropia, abordagens de Programação por Metas e técnicas multiobjetivo como o método ε-Constraint. A adoção desses métodos pode tornar a calibração dos pesos mais consistente e constitui uma oportunidade relevante para estudos futuros.

Outro ponto importante se refere ao tratamento de ordens de serviço emergenciais, que não foram consideradas explicitamente no modelo atual. Em cenários reais, a chegada súbita de uma ordem emergencial pode exigir reotimização imediata. Extensões naturais incluem a criação de um terceiro nível de prioridade (emergencial), o uso de janelas temporais críticas, mecanismos de preempção ou ainda a aplicação de modelos dinâmicos do tipo *rolling horizon*, capazes de reprocessar apenas as tarefas ainda não iniciadas e ajustar as rotas a partir da posição atual das equipes. Essas extensões tornam o modelo mais aderente ao ambiente operacional de campo e representam oportunidades importantes de continuidade do trabalho.

Para trabalhos futuros recomenda-se:

- Implementar um sistema de coleta de dados no Detran para, futuramente, realizar a comparação quantitativa do modelo proposto com o processo manual;
- Aplicar a Heurística Construtiva Gulosa e o VNS a outras meta-heurísticas não avaliadas neste trabalho;
- Expandir o modelo para um cenário dinâmico ou estocástico, que considere a chegada de novas ordens de serviço (prioritárias) ao longo do dia;

- Testar a escalabilidade das heurísticas em instâncias de grande porte (50 ou 100 tarefas).

O estudo realizado no Trabalho de Conclusão de Curso permitiu a aplicação prática dos conhecimentos adquiridos ao longo da graduação em Engenharia de Transportes e Logística, possibilitando o aprofundamento em otimização combinatória. Além disso, proporcionou a oportunidade de analisar um problema real e complexo do Detran Joinville, promovendo o desenvolvimento de habilidades como pensamento crítico e analítico. Essa experiência contribuiu para o fortalecimento de competências em modelagem matemática e implementação de meta-heurísticas, essenciais para a atuação profissional.

REFERÊNCIAS

ANAND, H.; AGGARWAL, V.; KUMAR, R. Optimization techniques for large scale problems. **International Journal of Operations Research**, v. 14, n. 3, p. 120–135, 2017.

APPLEGATE, D. L. et al. **The Traveling Salesman Problem: A Computational Study**. Princeton, New Jersey: Princeton University Press, 2007.

ARDIGO, J. D.; LUNA, H. P. L.; GOLDBARG, M. C. Um algoritmo branch-and-cut-and-price para o problema de roteamento de veículos com frota heterogênea. **Pesquisa Operacional**, v. 32, n. 3, p. 517–542, 2012.

ARENALES, M. N. et al. **Pesquisa Operacional**. São Paulo: Campus, 2007.

ASSAD, A. A. Modeling and implementation issues in vehicle routing. In: GOLDEN, B. L.; ASSAD, A. A. (Ed.). **Vehicle Routing: Methods and Studies**. North Holland, 1988. Disponível em: https://pdfs.semanticscholar.org/bd68/0d5708297eb76513611919c1145694d.

AZEVEDO, M. H. **Modelo para a alocação de recursos em sistemas de transporte**. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2010. Acesso em: 1 nov. 2024. Disponível em: http://objdig.ufrj.br/60/teses/coppe_m/MarcosHenriqueDeAzevedo.pdf.

BAKER, K. R. **Introduction to Sequencing and Scheduling**. New York: Wiley, 1974.

BALDACCI. A branch-and-cut algorithm for the vehicle routing problem with pickups and deliveries. **Transportation Science**, v. 40, n. 1, p. 5–19, 2006.

BELFIORE, P. P. **Scatter Search para problemas de roteirização de veículos com frota heterogênea, janelas de tempo e entregas fracionadas**. Tese (Doutorado) — Escola Politécnica da Universidade de São Paulo, São Paulo, 2005. Tese (Doutorado em Engenharia de Produção). Disponível em: http://www.teses.usp.br/teses/disponiveis/3/3136/tde-05092006-145756/pt-br.php.

BODIN, L. D. et al. Routing and scheduling of vehicles and crews: The state of the art. **Computers and Operations Research**, v. 10, n. 2, 1983.

BRANQUINHO, V. M. F. **Escalonamento de produção com tempos de setup dependentes – Aplicação na SAPEC Agro**. Dissertação (Dissertação (Mestrado em Decisão Económica e Empresarial)) — Lisbon School of Economics & Management, Lisboa, 2013.

BRASIL. **[Constituição (1988)]. Constituição da República Federativa do Brasil de 1988**. 1988. Brasília, DF: Presidência da República. Disponível em: http://www.planalto.gov.br/ccivil_03/constituicao/constituicao.htm. Acesso em: 27 out. 2025.

BUENO, F. **Métodos Heurísticos: Teoria e Implementações**. Araranguá: IFSC, 2017.

CHE, A.; CHU, C.; YANG, Y. Mixed integer programming formulations for parallel machine scheduling with release dates and sequence-dependent setup times. **Computers & Operations Research**, Elsevier, v. 83, p. 10–26, 2017.

Companhia de Engenharia de Tráfego CET. **Manual de sinalização viária. Volume 5: Sinalização horizontal**. 3. ed. São Paulo, 2019.

COSTA, M. A. B. Pesquisa operacional aplicada à agroindústria. In: BATALHA, M. O. (Ed.). **Gestão Agroindustrial**. São Paulo: Atlas, 1997. v. 2.

FEILLET, D. et al. A branch-and-price algorithm for the solution of the international newspaper problem. **INFORMS Journal on Computing**, v. 20, n. 3, p. 384–396, 2008.

FERNANDES, H. A. **Simulated annealing para programação de robôs móveis autônomos em ambiente de manufatura flexível**. Joinville: [s.n.], 2021. Trabalho de Conclusão de Curso (Graduação em Engenharia de Transportes e Logística) – Centro Tecnológico de Joinville, Universidade Federal de Santa Catarina.

FISHER, M. L.; JAIKUMAR, R. A generalized assignment heuristic for vehicle routing. **Networks**, v. 11, n. 2, p. 109–124, 1981.

FOURER, R.; GAY, D. M.; KERNIGHAN, B. W. **AMPL: A Modeling Language for Mathematical Programming**. 2. ed. [S.l.]: Brooks/Cole Publishing Company, 2003.

GARCIA, F.; BRANDO, L. T. **Pesquisa operacional: programação linear passo a passo, do entendimento do problema à interpretação da solução**. 2015. Recurso eletrônico. Disponível em: https://www.researchgate.net/publication/282974456_Pesquisa_operacional_programacao_linear_passo_a_passo_do_entendimento_do_problema_a_interpretacao_da_solucao_recurso_eletronico.

GIL, A. C. **Como elaborar projetos de pesquisa**. 4. ed. São Paulo: Atlas, 2002.

GLOVER, F.; KOCHENBERGER, G. A. **Handbook of Heuristics and Metaheuristics**. [S.l.]: Springer, 2003.

HILLIER, F. S.; LIEBERMAN, G. J. **Introdução à Pesquisa Operacional**. São Paulo: McGraw-Hill, 2013.

HOCHBAUM, D. S. **The Scheduling Problem**. 1999. RIOT: Remote Interactive Optimization Testbed. Disponível em: https://riot.ieor.berkeley.edu/Applications/Scheduling/l. Acesso em: novembro de 2022.

JOO, C. M.; KIM, B. S. Machine scheduling of time-dependent deteriorating jobs with determining the optimal number of rate modifying activities and the position of the activities. **Journal of Advanced Mechanical Design, Systems, and Manufacturing**, v. 9, n. 1, mar 2015.

KAILER, D. L. d. O.; TAGLIALENHA, S. L. d. S. Heuristic method for the twin robots scheduling problem in pickup and delivery stations. In: **Anais do LV Simpósio Brasileiro de Pesquisa Operacional (SBPO 2023)**. Local da Conferência, se souber: [s.n.], 2023. Disponível em: https://proceedings.science/sbpo/sbpo-2023/trabalhos/heuristic-method-for-the-twin-robots-scheduling-problem-in-pickup-and-delivery-s.

LACHTERMACHER, G. **Pesquisa Operacional na Tomada de Decisões**. 3. ed. Rio de Janeiro: LTC, 2009.

LAPORTE, G.; OSMAN, I. H. Local search in combinatorial optimization. In: AARTS, E.; LENSTRA, J. K. (Ed.). **Local Search in Combinatorial Optimization**. Chichester: Wiley, 1995.

LISBOA, E. **Apostila de Pesquisa Operacional**. 2002. Acesso em: 04 nov. 2024. Disponível em: https://ericolisboa.eng.br/cursos/apostilas/po/po.pdf.

LOUREIRO, N. F. P. **Utilização do simulated annealing na resolução de problemas no planejamento de produção**. Dissertação (Dissertação (Mestrado em Engenharia de Gestão Industrial)) — Faculdade de Ciências e Tecnologia da Universidade de Coimbra, Coimbra, 2014. Departamento de Engenharia Mecânica.

MARCONI, M. d. A.; LAKATOS, E. M. **Fundamentos de metodologia científica**. 5. ed. São Paulo: Atlas, 2003.

MLADENOVIC, N.; HANSEN, P. Variable neighborhood search. **Computers & Operations Research**, v. 24, p. 1097–1100, 1997.

NICHOLSON, T. **Optimization in industry: Optimization techniques**. London, UK: Longman Press, 1971. v. 1.

NOVAES, A. G. **Logística e Gerenciamento da Cadeia de Distribuição: Estratégia, Operação e Avaliação**. 2. ed. Rio de Janeiro: Elsevier, 2004.

PAPE, C. L. **Constraint-based scheduling: A tutorial**. 2015. [S.l.].

PINEDO, M. L. **Scheduling**. [S.l.]: Springer, 2012. v. 29.

PINEDO, M. L. **Scheduling: Theory, Algorithms, and Systems**. 5. ed. Cham: Springer, 2016.

REIS, J. **Uma introdução ao scheduling**. Lisboa: Instituto Superior de Ciências do Trabalho e da Empresa, 2006.

REKLAITIS, G. V. Review of scheduling of process operations. **Alche Symposium Series**, v. 78, n. 214, p. 119–133, 1982.

RIVAS, R. E. G. **Planejamento e controle da manutenção industrial: conceitos e aplicações**. 2009. Acesso em: 16 nov. 2024. Disponível em: http://www.pei.ufba.br/sites/pei.ufba.br/files/rene_ernesto_garcia_rivas.pdf.

RUSSELL, S.; NORVIG, P. **Inteligência Artificial**. 3. ed. Rio de Janeiro: Elsevier, 2013.

Senatran - Secretaria Nacional de Trânsito Contran. **Manual Brasileiro de Sinalização de Trânsito. Volume IV: Sinalização Horizontal**. 1. ed. Brasília, 2022.

SILVA, E. M. d. et al. **Pesquisa Operacional**. 1998. Acesso em: 3 nov. 2024. Disponível em: https://pt.scribd.com/doc/126242133/Ermes-Medeiros-Da-Silva-e-Outros-Pesquisa-Operacional.

TAGLIALENHA, S. L. d. S.; LáZARO, R. A. R. Electric transmission network expansion planning with the metaheuristic variable neighbourhood search. In: **Electric Transmission Network Expansion Planning with the Metaheuristic Variable Neighbourhood Search**. 1. ed. London: IntechOpen, 2019. v. 1, p. 568–667.

TAILLARD, É. D. **Design of Heuristic Algorithms for Hard Optimization**. Cham, Switzerland: Springer, 2023.

TALBI, E.-G. **Metaheuristics: From Design to Implementation**. [S.l.]: Wiley Publishing, 2009.

TEIXEIRA, d. C. **Análise de Estruturas de Vizinhaça: Um Estudo de Caso Sobre o Problema de Programação de Tarefas em Ambiente Job Shop**. Dissertação (Dissertação de Mestrado) — Universidade Federal da Bahia, Salvador, BA, Brasil, 2020. Acesso em: 09 nov. 2024. Disponível em: https://repositorio.ufba.br/handle/ri/33305?locale=en.

TOTH, P.; VIGO, D. **The vehicle routing problem**. Second. [S.l.]: SIAM, 2014. (Monographs on Discrete Mathematics and Applications).

VERGARA, S. C. **Projetos e relatórios de pesquisa em administração**. 10. ed. São Paulo: Atlas, 2009.

VIEIRA, R. C.; SENA TAGLIALENHA, S. L. de. Scheduling autonomous mobile robots in flexible manufacturing environment using neighborhood search metaheuristics. In: **Anais do LV Simpósio Brasileiro de Pesquisa Operacional**. São José dos Campos: Galoá, 2023. Disponível em: https://proceedings.science/sbpo/sbpo-2023/trabalhos/scheduling-autonomous-mobile-robots-in-flexible-manufacturing-environment-using?lang=pt-br. Acesso em: 29 out. 2025. Disponível em: https://proceedings.science/sbpo/sbpo-2023/trabalhos/scheduling-autonomous-mobile-robots-in-flexible-manufacturing-environment-using?lang=pt-br.

APÊNDICE A - Código do Modelo Exato

```
import gurobipy as gp
from gurobipy import GRB
import random

n_tarefas = 15
NOS = ['Deposito'] + [f'OS{i}' for i in range(1, n_tarefas + 1)]
TAREFAS = [f'OS{i}' for i in range(1, n_tarefas + 1)]
DEPOSITO = ['Deposito']
O = ['EQUIPE1', 'EQUIPE2', 'EQUIPE3']
H = {'EQUIPE1': 480, 'EQUIPE2': 480, 'EQUIPE3': 480}

prioridade_sim = [1, 8, 15]
PRIORIDADE = {'Deposito': 0}
for i in range(1, n_tarefas + 1):
    PRIORIDADE[f'OS{i}'] = 1 if i in prioridade_sim else 0

m2_data = [20, 15, 2, 5, 22, 2, 4, 5, 6, 1, 2, 10, 6, 4, 1]
AREA = {'Deposito': 0}
for i in range(1, n_tarefas + 1):
    AREA[f'OS{i}'] = m2_data[i-1]

S = {'Deposito': 0}; C = {'Deposito': 0}
for i in range(1, n_tarefas + 1):
    S[f'OS{i}'] = 10
    C[f'OS{i}'] = 10

tempo_processamento = {i: (S[i] + AREA[i] * 10 + C[i]) for i in
   TAREFAS}

prazos = {}
for tarefa in TAREFAS:
    dias_aleatorios = random.randint(1, 3)
```

```
    prazos[tarefa] = dias_aleatorios * 480
print("--- Prazos Gerados (em minutos) ---\n", prazos, "\n" + "-" *
    35)

T_data = [
    [0.0, 12.0, 8.0, 17.0, 11.0, 9.0, 8.0, 5.0, 5.0, 8.0, 6.0, 6.0,
        11.0, 12.0, 5.0, 14.0],
    [13.0, 0.0, 7.0, 12.0, 5.0, 13.0, 17.0, 12.0, 15.0, 16.0, 13.0,
        11.0, 10.0, 12.0, 15.0, 11.0],
    [9.0, 7.0, 0.0, 14.0, 5.0, 8.0, 11.0, 6.0, 11.0, 12.0, 7.0,
       8.0, 6.0, 10.0, 11.0, 5.0],
    [19.0, 11.0, 13.0, 0.0, 11.0, 11.0, 12.0, 14.0, 15.0, 11.0,
       17.0, 19.0, 20.0, 7.0, 14.0, 5.0],
    [11.0, 5.0, 3.0, 13.0, 0.0, 9.0, 12.0, 7.0, 13.0, 16.0, 8.0,
       11.0, 10.0, 8.0, 12.0, 10.0],
    [9.0, 12.0, 7.0, 11.0, 9.0, 0.0, 6.0, 5.0, 8.0, 13.0, 2.0,
       12.0, 12.0, 5.0, 7.0, 8.0],
    [9.0, 16.0, 11.0, 12.0, 13.0, 4.0, 0.0, 8.0, 5.0, 9.0, 5.0,
       12.0, 16.0, 6.0, 3.0, 9.0],
    [8.0, 10.0, 5.0, 11.0, 7.0, 5.0, 8.0, 0.0, 9.0, 13.0, 4.0,
       11.0, 10.0, 6.0, 8.0, 9.0],
    [5.0, 14.0, 11.0, 17.0, 14.0, 9.0, 7.0, 8.0, 0.0, 6.0, 6.0,
       7.0, 14.0, 12.0, 4.0, 13.0],
    [8.0, 15.0, 12.0, 17.0, 15.0, 11.0, 9.0, 14.0, 6.0, 0.0, 10.0,
       8.0, 15.0, 13.0, 8.0, 14.0],
    [7.0, 13.0, 7.0, 12.0, 10.0, 3.0, 5.0, 4.0, 6.0, 11.0, 0.0,
       10.0, 13.0, 6.0, 4.0, 9.0],
    [6.0, 13.0, 10.0, 22.0, 13.0, 13.0, 12.0, 11.0, 8.0, 10.0, 0.0,
        13.0, 16.0, 9.0, 19.0, 0.0],
    [10.0, 10.0, 5.0, 19.0, 9.0, 12.0, 14.0, 9.0, 13.0, 14.0, 10.0,
        9.0, 0.0, 15.0, 12.0, 16.0],
    [12.0, 12.0, 11.0, 7.0, 11.0, 6.0, 6.0, 9.0, 10.0, 12.0, 8.0,
       17.0, 16.0, 0.0, 9.0, 4.0],
    [14.0, 14.0, 9.0, 13.0, 12.0, 5.0, 5.0, 6.0, 5.0, 10.0, 1.0,
       9.0, 13.0, 8.0, 0.0, 11.0],
    [15.0, 10.0, 10.0, 5.0, 9.0, 7.0, 9.0, 10.0, 13.0, 15.0, 9.0,
       18.0, 17.0, 4.0, 12.0, 0.0]
]
T = {(NOS[i], NOS[j]): T_data[i][j] for i in range(len(NOS)) for j
   in range(len(NOS))}
```

```
try:
    m = gp.Model("modelo_completo_flexivel")
    m.setParam('TimeLimit', 120)
    M = 9999

    y = m.addVars(NOS, NOS, O, vtype=GRB.BINARY, name="y")
    z = m.addVars(NOS, O, vtype=GRB.BINARY, name="z")
    x = m.addVars(TAREFAS, O, vtype=GRB.CONTINUOUS, name="x")
    u = m.addVars(TAREFAS, lb=1.0, ub=n_tarefas, vtype=GRB.
       CONTINUOUS, name="u")
    s_plus = m.addVars(TAREFAS, O, vtype=GRB.CONTINUOUS, name="
       s_plus")
    s_minus = m.addVars(TAREFAS, O, vtype=GRB.CONTINUOUS, name="
       s_minus")

    custo_viagem = 0.6 * gp.quicksum(T[i,j] * y[i,j,k] for i in
       TAREFAS for j in NOS for k in O)
    bonus_tarefas = gp.quicksum((40 + 20 * PRIORIDADE[i]) * z[i,k]
       for i in TAREFAS for k in O)
    custo_multa = 0.4 * gp.quicksum(s_plus[j,k] for j in TAREFAS
       for k in O)
    m.setObjective(custo_viagem - bonus_tarefas + custo_multa, GRB.
       MINIMIZE)

    for k in O:
        tempo_viagem = gp.quicksum(T[i,j] * y[i,j,k] for i in NOS
           for j in NOS)
        tempo_servico = gp.quicksum(tempo_processamento[i] * z[i,k]
            for i in TAREFAS)
        m.addConstr(tempo_viagem + tempo_servico <= H[k], name=f"
           Jornada_Maxima_{k}")

    for i in TAREFAS:
        m.addConstr(gp.quicksum(z[i,k] for k in O) <= 1, name=f"
           Atribuicao_Unica_{i}")

    for i in TAREFAS:
        for k in O:
            m.addConstr(gp.quicksum(y[j,i,k] for j in NOS) == z[i,k
               ], name=f"Fluxo_Entrada_{i}_{k}")
```

```
        m.addConstr(gp.quicksum(y[i,j,k] for j in NOS) == z[i,k
            ], name=f"Fluxo_Saida_{i}_{k}")

for k in O:
    saida_deposito = gp.quicksum(y[d,j,k] for d in DEPOSITO for
        j in TAREFAS)
    retorno_deposito = gp.quicksum(y[i,d,k] for i in TAREFAS
       for d in DEPOSITO)
    m.addConstr(saida_deposito <= 1, name=f"Saida_Unica_{k}")
    m.addConstr(saida_deposito == retorno_deposito, name=f"
       Equilibrio_Deposito_{k}")

for k in O:
    for i in TAREFAS:
        m.addConstr(x[i,k] >= gp.quicksum(T[d,i] * y[d,i,k] for
            d in DEPOSITO), name=f"Inicio_Primeira_Tarefa_{i}_{
           k}")
        for j in TAREFAS:
            if i != j:
                m.addConstr(x[j,k] >= x[i,k] +
                   tempo_processamento[i] + T[i,j] - M * (1 - y
                   [i,j,k]), name=f"Sequenciamento_{i}_{j}_{k}"
                   )

for i in TAREFAS:
    for k in O:
        m.addConstr(x[i,k] <= M * z[i,k], name=f"
           Vinculo_Tempo_Atribuicao_{i}_{k}")

for j in TAREFAS:
    for k in O:
        m.addConstr(x[j,k] + tempo_processamento[j] - prazos[j]
            <= s_plus[j,k] - s_minus[j,k] + M * (1-z[j,k]),
           name=f"Prazo_{j}_{k}")

m.addConstrs((y[i,i,k] == 0 for i in NOS for k in O), name="
   Proibir_Self_Loops")

for i in TAREFAS:
    for j in TAREFAS:
        if i != j:
```

```
                for k in O:
                    m.addConstr(u[i] - u[j] + n_tarefas * y[i,j,k]
                        <= n_tarefas - 1, name=f"Eliminacao_Subrota_
                        {i}_{j}_{k}")

m.optimize()

if m.status == GRB.OPTIMAL or m.status == GRB.TIME_LIMIT:
    print("\n################ RESULTADOS ################")
    print(f"Valor da Fun   o Objetivo: {m.objVal:.2f}\n")

    tarefas_realizadas_total = 0
    for i in TAREFAS:
        for k in O:
            if z[i,k].X > 0.5:
                tarefas_realizadas_total += 1
    print(f"--- Total de Tarefas Realizadas: {
       tarefas_realizadas_total} de {n_tarefas} ---\n")

    print("--- Tempos de In cio e Atrasos ---")
    for i in TAREFAS:
        for k in O:
            if z[i,k].X > 0.5:
                print(f"Tarefa {i} (Equipe {k}): In cio={x[i,k
                    ].X:.2f}, Atraso={s_plus[i,k].X:.2f} min (
                    Prazo: {prazos[i]})")

    print("\n--- Rotas Completas ---")
    for k in O:
        ponto_partida = None
        for j in TAREFAS:
            if y[DEPOSITO[0], j, k].X > 0.5:
                ponto_partida = j
                break

        if ponto_partida:
            rota_str = f"EQUIPE {k}: Deposito -> {ponto_partida
                }"
            posicao_atual = ponto_partida
            for _ in range(n_tarefas):
                proximo_ponto = None
```

```
                    for j in TAREFAS:
                        if y[posicao_atual, j, k].X > 0.5:
                            proximo_ponto = j
                            break
                    if proximo_ponto:
                        rota_str += f" -> {proximo_ponto}";
                           posicao_atual = proximo_ponto
                    else: break
                rota_str += " -> Deposito"; print(rota_str)
            else:
                print(f"EQUIPE {k}: Nenhuma rota.")
    else:
        print("N o foi encontrada uma solu  o  tima .")
        if m.status == GRB.INFEASIBLE:
            print("O modelo    invi vel. Verificando as
               restri  es conflitantes..."); m.computeIIS()
            m.write("modelo_inviavel.ilp")
            print("\nUm arquivo 'modelo_inviavel.ilp' foi criado.
               As restri  es listadas nele s o a causa da
               inviabilidade.")

except gp.GurobiError as e:
    print(f"Erro do Gurobi: {e}")
except Exception as e:
    print(f"Ocorreu um erro no Python: {e}")
```