UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO DE JOINVILLE
CURSO DE ENGENHARIA NAVAL

LETÍCIA DE MELLO OLIVEIRA DE SOUZA

ROTEIRIZAÇÃO DE VEÍCULOS PARA ABASTECIMENTO DE LINHAS DE MONTAGEM

Joinville
2023

LETÍCIA DE MELLO OLIVEIRA DE SOUZA

ROTEIRIZAÇÃO DE VEÍCULOS PARA ABASTECIMENTO DE LINHAS DE MONTAGEM

Trabalho apresentado como requisito para obtenção do título de bacharel em Engenharia Naval do Centro Tecnológico de Joinville da Universidade Federal de Santa Catarina.

Orientadora: Dra. Eng. Vanina Macowski Durski Silva

Joinville
2023

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Souza, Letícia de Mello
ROTEIRIZAÇÃO DE VEÍCULOS PARA ABASTECIMENTO DE LINHAS DE MONTAGEM / Letícia de Mello Souza ; orientadora, Vanina Macowski Durski Silva, 2023.
107 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Campus Joinville, Graduação em Engenharia Naval, Joinville, 2023.

Inclui referências.

1. Engenharia Naval. 2. Roteirização. 3. Otimização de rotas. 4. Linhas de montagem. 5. Logística de abastecimento. I. Silva, Vanina Macowski Durski. II. Universidade Federal de Santa Catarina. Graduação em Engenharia Naval. III. Título.

LETÍCIA DE MELLO OLIVEIRA DE SOUZA

ROTEIRIZAÇÃO DE VEÍCULOS PARA ABASTECIMENTO DE LINHAS DE MONTAGEM

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

Joinville (SC), 05 de Dezembro de 2023.

Banca Examinadora:

Documento assinado digitalmente
Vanina Macowski Durski Silva
Data: 13/12/2023 19:36:06-0300
CPF: ***.806.619-**
Verifique as assinaturas em https://v.ufsc.br

_____

Orientadora: Dra. Eng. Vanina Macowski Durski Silva
Orientadora
Presidente

_____

Dra. Eng. Christiane Wenck Nogueira Fernandes
Membro
Universidade Federal de Santa Catarina

_____

Dra. Eng. Francielly Hedler Staudt
Membro
Universidade Federal de Santa Catarina

AGRADECIMENTOS

Gostaria de expressar minha profunda gratidão à minha amada família, em especial à minha mãe e à minha avó, cujo apoio me guiou ao longo da minha jornada acadêmica e profissional. Agradeço por todo o esforço dedicado a me fazer enxergar a vida de maneira mais tranquila e relaxada. Com carinho, dedicação e confiança, vocês moldaram a pessoa que sou hoje. A vocês, dedico toda minha gratidão.

Estendo meus agradecimentos aos amigos que caminharam ao meu lado durante toda a graduação: Laura, Samuel e Mikha. Samuel, uma pessoa tão diferente de mim e, ao mesmo tempo, tão parecida, tornou-se inestimável em minha vida em pouco tempo. Agradeço a Laura, que compartilhou momentos alegres e difíceis ao meu lado, sendo fundamental para que essa jornada fosse possível. Obrigada por sempre acreditar no meu melhor, me apoiar e nunca sair do meu lado! À Mikha, agradeço por toda a calma que sempre transmitiu ao longo do curso, por todos os trabalhos e momentos que compartilhamos juntas. Muito obrigada por tudo. O apoio e a amizade de vocês foram pilares essenciais ao longo dessa trajetória.

Agradeço imensamente à minha orientadora, Vanina, pela sua fundamental contribuição no meu percurso acadêmico. Seu papel não se limitou apenas à orientação, mas foi marcado pela crença constante em meu potencial, pelo estímulo na busca pelo conhecimento e pelas experiências compartilhadas que enriqueceram significativamente minha jornada. Esses ensinamentos não apenas moldaram minha visão profissional, mas também contribuíram de maneira crucial para meu desenvolvimento pessoal. Sua disposição para ajudar, paciência notável e apoio contínuo foram pilares essenciais ao longo dessa jornada, e por isso, expresso minha sincera gratidão. Gostaria de estender meus agradecimentos, também, ao Gustavo Costa, que desempenhou um papel significativo em parte da minha jornada acadêmica. Sua paciência e o incentivo para me introduzir no mundo da simulação foram elementos importantes para o meu progresso e aprendizado ao longo desses anos na faculdade.

Por fim, dedico meus sinceros agradecimentos ao corpo docente da Universidade Federal de Santa Catarina, em especial ao professor Thiago Pontin, cujos conhecimentos e aulas em engenharia naval foram verdadeiramente primorosos. Seus ensinamentos sobre o desenvolvimento do senso crítico de um engenheiro e a aplicação prática desses princípios na engenharia naval foram inestimáveis para minha formação. Além disso, expresso minha gratidão à professora Francielly pela paciência e dedicação em guiar pelo mundo da logística.

RESUMO

As cadeias de abastecimento possuem processos relevantes no que tange à organização das linhas de montagem de indústrias automotivas. Nesse processo, o qual consiste em distribuir, armazenar e organizar as peças do supermercado ao longo da linha de montagem, objetiva-se garantir a otimização da cadeia produtiva a fim de reduzir custos e aumentar a competitividade das empresas. Em virtude da variação na frequência de demanda dos diferentes componentes inclusos nas linhas de montagem mistas, a má gestão do processo de abastecimento torna-se uma problemática de destaque à medida em que os movimentos de distribuição e organização não otimizados tendem a custar tempo extra e possíveis paradas na produção, significando desperdício de recursos. A literatura sobre tal tema, entretanto, ainda é escassa e, grande parte das soluções de roteirização empregadas na indústria se baseiam na experiência dos operadores frente a estudos específicos de distribuição. Propõe-se, portanto, um estudo acerca dos métodos ótimos do fluxo de material de supermercados para as linhas de montagem, reduzindo os custos e o tempo do movimento de reabastecimento. Neste estudo, os materiais a serem transportados são armazenados em um local de estoque e devem ser transferidos para pontos distribuídos ao longo da linha de montagem por meio de *tow trains* com capacidade máxima fixa. Propõe-se para a resolução do problema de roteirização, uma ferramenta computacional heurística para o planejamento de rotas de *tow trains*, a qual foi implementada com a linguagem de programação Python e integrou a biblioteca OR-Tools. Ao algoritmo foram impostas diversas condicionais e restrições, incluindo restrições de demanda e capacidade específicas, direções no *line supply*, janelas de tempo de entregas e condicionais relacionadas ao reabastecimento nos supermercados. Os resultados do estudo demonstram que o algoritmo desenvolvido obteve sucesso em otimizar as operações de reabastecimento em linhas de montagem automotivas, atingindo valores ótimos de distâncias percorridas, equivalentes a 10.325 metros total por rota durante um turno de trabalho e uma frota ótima equivalente a 5 *tow trains*. Destaca-se um importante *trade off* nas mudanças de janelas de tempo e no número de veículos, sendo que as mudanças de janelas de tempo podem causar flutuações de cerca de 12,5% a 18,75% no atendimento da demanda, enquanto a redução na frota de veículos pode impactar em até 43,75% tal atendimento. Neste ponto, o algoritmo permitiu a constatação de que o cenário mais eficiente é o representado pela matriz de janelas de tempo M4 sem restrições no *line supply*, com cinco *tow trains*. Por fim, a flexibilidade do algoritmo sugere aplicabilidade em diversos contextos logísticos, evidenciando seu equilíbrio entre complexidade e eficiência na roteirização de *tow trains*.

Palavras-chave: Roteirização. Logística de abastecimento. Otimização de rotas. Linhas de montagem.

ABSTRACT

Supply chains are pivotal processes in the organization of assembly lines within the automotive industry. In this process, which consists of distributing, storing and organizing supermarket parts along the assembly line, the aim is to guarantee the optimization of the production chain in order to reduce costs and increase the competitiveness of companies. Due to the varying demand frequency of components in mixed assembly lines, mismanagement of the supply process becomes a significant issue as non-optimized distribution and organization movements tend to cost extra time and possible stops in production, meaning waste of resources. Literature on this topic remains scarce, and many routing solutions employed in the industry rely on operators' experiences in specific distribution studies. Therefore, a study is proposed to investigate optimal methods for supermarket material flow to assembly lines, with the goal of reducing costs and replenishment movement time. In this study, materials to be transported are stored in a designated stock location and must be transferred to points distributed along the assembly line through tow trains with a fixed maximum capacity. To address the routing problem, a heuristic computational tool for tow train route planning is proposed. It is implemented using the Python programming language and integrated with the OR-Tools library. The algorithm is subject to various conditions and constraints, including specific demand and capacity restrictions, directions in the line supply, delivery time windows, and conditions related to replenishment in supermarkets. The results of the study demonstrate that the developed algorithm successfully optimized replenishment operations in automotive assembly lines. It achieved optimal values for distances traveled, equivalent to a total of 10.325 meters per route during a work shift, and an optimal fleet equivalent to 5 tow trains. A significant trade-off is highlighted in changes to time windows and the number of vehicles. Alterations in time windows can cause fluctuations of approximately 12,5% to 18,75% in demand fulfillment, while reducing the vehicle fleet can impact this fulfillment by up to 43,75%. At this point, the algorithm allowed the identification that the most efficient scenario is represented by the time window matrix M4 without line supply restrictions, with five tow trains. Finally, the flexibility of the algorithm suggests applicability in various logistical contexts, highlighting its balance between complexity and efficiency in tow train routing.

Keywords: Routing. Logistics. Optimization. Assembly lines.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

MMAL	Linhas de montagem mistas
VRP	Problema de roteirização de veículos
TSP	Problema do caixeiro viajante
AE	Algoritmos exatos
AA	Algoritmos aproximados
SOG	Soluções ótimas globais
SOL	Soluções ótimas locais
MTS	*Make to Stock*
ATO	*Assemble to Order*
MTO	*Make to Order*
ETO	*Engineer to Order*
GRASP	*Greedy Randomized Adaptive Search Procedure*
SA	*Simulated annealing*
VNS	*Variable neighborhood search*
TE	Tempo de Operação
QS	Qualidade da solução
TTU	Número de *Tow Trains* Utilizados
VMS	*Virtual Memory Size*
RSS	*Resident Set Size*
RAM	Memória física
SWP	Memória de troca
GLS	*Guided Local Search*

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

No final do século XIX, as indústrias automotivas estadunidenses foram diretamente influenciadas pelo sistema de produção Taylorista-Fordista. Esse sistema foi definido pela presença de linhas de montagem, onde os funcionários realizavam tarefas padronizadas de produção repetitiva, que exigiam qualificações mínimas em tempo controlado, desenvolvendo produtos com pouca variedade (ALI, 2019).

Nas organizações, um maior grau de eficiência e competência é atingido quando, diariamente, consegue-se o maior rendimento possível dos recursos (TAYLOR, 1990). Tal citação, utilizada no contexto da 1º Revolução Industrial, revela um pensamento comum à ótica capitalista: a obtenção de lucro baseada na redução do tempo de produção.

No modelo capitalista de produção, os acréscimos de produtividade obtidos com a padronização eram correspondidos por benefícios salariais, o que proporcionou uma distribuição de renda equitativa, sendo que a massa de salários completava, na outra ponta, o consumo de massa. Infere-se, a partir de tal, que, à medida que cientificava a administração das empresas, o Taylorismo-Fordismo consolidou a otimização produtiva das linhas de montagem como uma prioridade nos campos fabris (LEMOS, 1991).

Ainda que a visão de Taylor tenha sido corroborada pelo sucesso das indústrias automotivas, sobretudo a americana no final do século XIX, essa foi substituída pelo sistema de produção de manufatura enxuta (DEGAN, 2011). Esse novo fluxo impulsiona a necessidade das empresas fabricarem diferentes produtos em uma sequência influenciada pela demanda de pedidos dos clientes, seguindo a ótica Just-In-Time (JIT) (SONG; ZIPKIN, 2003).

Com essa transição, as empresas comumente associadas à fabricação de produtos únicos, foram substituídas por linhas de montagem mistas (MMALs). Assim, diversos modelos de produtos são fabricados em uma mesma linha de montagem, proporcionando maior flexibilidade de produção e, consequentemente, maior competitividade no mercado (ALI, 2019).

Esse modelo de produção, caracterizado pela ampla variedade de produtos fabricados, ainda que apresente vantagem competitiva para as montadoras de automóveis modernas, gera uma problemática no que tange à distribuição de peças na linha de produção. Devido ao espaço limitado nas linhas, a complexidade de operação em uma linha mista, com várias peças e componentes circulando no chão de fábrica, tendem a provocar conflitos, caso não seja adequadamente administrado (CUNHA; WU, 2008).

Nesse contexto, os supermercados apresentam-se como uma alternativa ao depósito de componentes na linha de montagem, ao passo que atuam como pontos de alocação para as diversas peças que virão a integrar os produtos na linha de montagem. De acordo com os autores, esses locais são caracterizados por serem uma área logística descentralizada do chão de fábrica das montadoras, onde são armazenadas as peças que serão enviadas às estações de trabalho (BATINNI; BOYSEN; EMDE, 2013).

Diferentes estações de trabalho podem ser atendidas por um único supermercado, e o chão de fábrica das montadoras pode ter vários supermercados instalados. A logística de abastecimento desses supermercados, é caracterizada pela chegada dos componentes nos armazéns, seguida pela separação para posterior organização, de acordo com as demandas das estações de trabalho localizadas ao longo da linha de montagem (ALI, 2019).

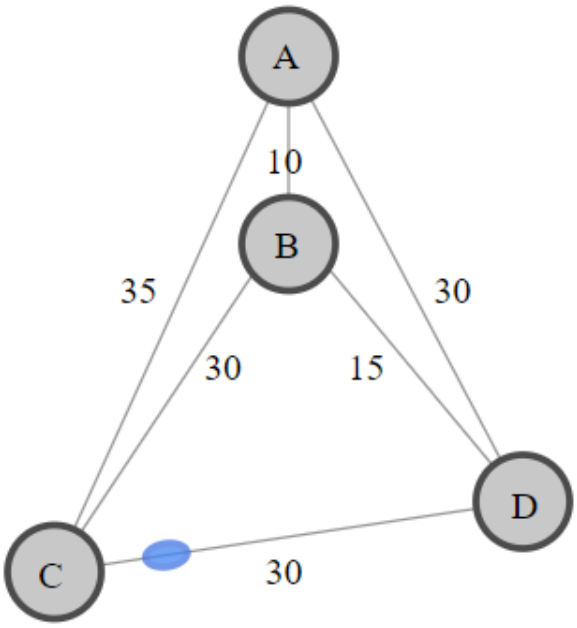
Conforme a demanda por peças, essas podem ser transportadas do supermercado até as estações de trabalho por meio de reboques (*tow trains*). Tem-se então, o processo de abastecimento da linha de montagem atrelado ao planejamento das rotas e o *schedule* a serem realizados pelos *tow trains*, de modo que as estações de trabalho sejam atendidas sem a necessidade de parar a linha de montagem, ou sem gerar um excesso de peças paradas na linha (BATTINI *et al.*, 2013).

O abastecimento das linhas de montagem é um processo fundamental para o sucesso operacional das montadoras. De acordo com Sule (2009), as atividades de movimentação podem superar 30% dos custos operacionais totais de uma planta de manufatura e economias da ordem de 15% a 20% podem ser alcançadas mediante uma revisão do processo. Em função disso, constata-se a necessidade de compreender o processo de roteirização dos veículos envolvidos no reabastecimento de peças em linhas de montagem, mais especificamente dos reboques, nos casos das montadoras de veículos.

Os cenários de roteirização de veículos (VRP) foram, primeiramente, formulados por Dantzig e Ramser (1959), que os definem como uma generalização do problema do caixeiro viajante (TSP). Objetiva-se, no TSP, encontrar o trajeto mais curto – seja em distância total percorrida ou tempo de viagem – em que o caixeiro consiga visitar os clientes em locais distintos, voltando, por fim, ao local inicial de sua rota.

A roteirização do percurso do caixeiro pode ser representada graficamente por nós e arcos que permitem definir o deslocamento. A Figura 1 apresenta um grafo contendo quatro pontos, cujas distâncias entre si são expressas sob as arestas, indicando possibilidade de realização de viagem entre um ponto e outro. Neste exemplo, ao presumir-se o início do percurso no nó A, aponta-se que a distância ótima equivale a 90 u.m, sendo essa correspondente ao trajeto “ACDBA”.

Figura 1 – Problema com o vendedor viajante



Fonte: Elaboração própria (2023).

Ainda que a solução iterativa aparente simplicidade ao alocar um número reduzido de nós, o problema torna-se complexo com a adição de pontos durante o processo de otimização (CUNHA; WU, 2008). Em paralelo, situações condicionais podem ser adicionadas a tais problemáticas, como a definição do volume máximo de itens a serem transportados pelos rebocadores ou horários específicos para a realização das rotas. A partir disso, o processo iterativo para obtenção da solução ótima cresce exponencialmente, fato que aumenta o custo computacional (JOURDAN; BASSEUR; TALBI, 2009).

Aponta-se que os métodos de roteirização computacionais podem ser substituídos por determinações manuais de rotas baseadas na experiência dos responsáveis pela logística de abastecimento das montadoras. Entretanto, tais soluções podem diferir da solução ótima, apresentar desbalanceamento na atuação dos rebocadores ou gerar problemas na alocação das peças, uma vez que não se baseiam em um método comprobatório, e sim, em pressupostos dos operadores. Desse modo, é imprescindível, em problemas maiores, utilizar técnicas de otimização que abordem a roteirização a partir de métodos heurísticos (CUNHA; WU, 2008).

Assim, neste estudo propõe-se o desenvolvimento de um algoritmo que sistematize o processo iterativo da rotina de otimização das rotas realizadas pelos *tow trains* durante o transporte de peças do supermercado até as estações de trabalho ao longo da linha de montagem. Como as rotas realizadas na linha de montagem são

definidas a partir de características específicas das peças transportadas, espera-se que o algoritmo empregue condicionais as quais se apliquem ao processo de uma montadora de automóveis.

A melhoria de rotas levará em conta o número de *tow trains* nas linhas de montagem, a movimentação total dos carrinhos (*trolleys*) e a dinâmica de tempo que configure um padrão dinâmico de transporte de peças. Isso tende a reduzir os custos de produção devido a eficiência nos movimentos de reabastecimento da linha de montagem e a alavancar a eficiência produtiva em função da previsão com antecedência da localização das peças nas estações de trabalho.

1.2 OBJETIVOS

Pondo-se em evidência a necessidade de garantir a eficiência das cadeias de abastecimento em um ambiente fabril de linhas de montagem automotivas, propõe-se os seguintes objetivos.

1.2.1 Objetivo Geral

Propor uma ferramenta computacional heurística de planejamento de rotas de *tow trains* para reabastecimento de peças em linhas de montagem automotiva.

1.2.2 Objetivos Específicos

- Realizar levantamento bibliográfico sobre modelos de roteirização utilizados em problemas de reabastecimento de linhas de montagem;
- Desenvolver um algoritmo para redução das distâncias totais viajadas pelos *tow trains*, bem como, do número total de *tow trains* utilizados para realizar as entregas previstas;
- Realizar análises de indicadores (número de *tow trains* empregados para realização das entregas, distância total percorrida na rota, otimização das janelas de tempo), gerando recomendações para otimizar o processo de roteamento e *scheduling*.

1.3 JUSTIFICATIVAS DO TRABALHO

1.3.1 Econômica e Operacional

Em indústrias automobilísticas, caracterizadas por possuírem linhas contínuas de montagem ou fabricação é indispensável garantir um fornecimento de peças de modo confiável e flexível, a fim de evitar paralisações na linha e a consequente perda de lucratividade. Nessas fábricas, diferentes peças devem ser transferidas entre uma área de estoque (supermercado) até as estações de trabalho dispostas ao longo das

linhas de montagem, processo que exige um cuidadoso planejamento (CUNHA; WU, 2008).

O processo de reabastecimento das linhas é realizado com o auxílio dos *tow trains*, um veículo no qual são atrelados múltiplos *trolleys* contendo as peças solicitadas pelas estações. Esse transporte conta, ainda, com desafios complexos como a necessidade de garantir o atendimento de janelas de tempo de entrega, a capacidade de transporte de cada *tow train* e *trolley* ou a realização de manobras com os veículos (ALI, 2019).

Desse modo, comprova-se que a roteirização e *scheduling* de *tow trains* é um importante problema de otimização para as montadoras de automóveis. Ademais, destaca-se a importância do planejamento adequado de tais atividades, em função de seu alto custo operacional - (SULE, 2009) afirma que as atividades de movimentação podem superar 30% dos custos operacionais totais de uma planta.

A implementação computacional de um algoritmo capaz de abordar todas essas condicionais de forma integrada para a resolução de um problema de roteirização e *scheduling* em linhas de montagem, então, pode resultar em economia substancial de recursos, redução de despesas operacionais e melhoria da margem de lucro.

As vantagens econômicas e operacionais de desenvolver tal código computacional são notáveis à medida que, ao agregar várias restrições complexas e representativas do ambiente operacional das montadoras em um único código, as indústrias automotivas podem otimizar suas rotas de transporte e distribuição de maneira mais eficaz. Isso resulta em redução de custos de combustível/energia, menor desgaste de veículos e aumento da produtividade da mão de obra. Além disso, contribui com o atendimento das demandas dos clientes de forma mais eficiente melhora a satisfação do cliente e pode abrir oportunidades para conquistar novos negócios.

Operacionalmente, a implementação desse código permite uma melhor alocação de recursos, reduzindo os tempos de espera, minimizando os atrasos nas entregas e simplificando o planejamento logístico. Isso leva a uma operação mais fluída e eficiente, reduzindo os gargalos na linha de montagem e permitindo que as empresas atendam às demandas de produção de forma consistente.

Em resumo, a criação de um código computacional para resolução de problemas de roteirização e *scheduling* em linhas de montagem não apenas proporciona vantagens econômicas substanciais, como a redução de custos e o aumento da eficiência, mas também oferece benefícios operacionais significativos, incluindo melhorias na gestão de recursos e na satisfação do cliente.

1.3.2 Acadêmica

Através de inspeções realizadas durante o desenvolvimento do presente estudo, constatou-se que muitos dos problemas de roteirização envolvem a adição de múltiplas

restrições como janelas de tempo de entregas, restrições de capacidade de *tow trains* e *trolleys*, matrizes de distância variáveis, limitações de idas e vindas na linha de montagem, entre outras restrições que impõem complexidade ao problema (KRUK, 2018). Ainda assim, em virtude de tal complexidade oriunda da integração de problemas, destaca-se que a literatura existente, frequentemente, tende a abordar as restrições de forma isolada, o que pode limitar a aplicabilidade prática das soluções propostas (ARAUJO, 2001).

Com isso, infere-se que este trabalho apresenta a proposta e o desenvolvimento de um código computacional para a resolução de problemas de roteirização com integração de restrições pouco exploradas (como janelas de tempo e restrições no *line supply*), o que indica um potencial de contribuir no âmbito acadêmico, apresentando o equacionamento matemático que rege a sobreposição destas condições, bem como a esquematização de um algoritmo para a solução simplificada de problemas de roteirização com diferentes configurações.

A aplicação da ferramenta OR-Tools, aqui empregada para adição de sua biblioteca própria de roteirização ao código a ser desenvolvido, também apresenta um potencial pouco explorado, em função do método aplicado para criação do algoritmo. Apesar de existirem publicações que relatam a aplicação da biblioteca para resolução de problemas simples de roteirização, a adição de condicionais em paralelo torna sua utilização inviável por meio de seu código original (ALDEANO, 2020). Com isso, o presente trabalho utiliza uma abordagem metodológica de integração da biblioteca de roteirização do OR-Tools com a solução numérica básica, adaptadas para a construção de um algoritmo em Python.

Conclui-se, assim, que o desenvolvimento do presente trabalho sobre a criação de um código computacional para a resolução de problemas de roteirização em linhas de montagem não apenas aborda desafios práticos significativos na indústria, mas também contribui para o avanço do conhecimento acadêmico à medida que propõe-se a expandir as soluções de roteirização - por meio da agregação de múltiplas condicionais aplicadas na indústria montadora de automóveis.

Pode-se destacar, ainda, que a criação do algoritmo não está limitada a uma única disciplina acadêmica, visto que abrange campos como a otimização, a engenharia de produção, a logística e até mesmo a engenharia naval. Essa abordagem interdisciplinar é valiosa, pois permite que profissionais e acadêmicos de várias áreas colaborem e utilizem o código em uma variedade de contextos. No caso da engenharia naval a otimização de rotas e a gestão eficiente de recursos são essenciais para a logística de suprimentos em estaleiros e portos.

Outra vantagem acadêmica do código para a engenharia naval é a sua aplicação para otimização de rotas de veículos, podendo ser aplicado para roteirizar equipamentos de transporte de cargas em portos e *depots*, reduzindo gastos com

movimentações desnecessárias e emissões de poluentes. Ao contribuir para o desenvolvimento de soluções eficazes em logística e roteirização, o trabalho acadêmico pode indiretamente melhorar a eficiência e a competitividade da indústria naval.

1.4 ESTRUTURA DO TRABALHO

Destaca-se que o presente trabalho está organizado da seguinte maneira:

O capítulo 1 conta com a introdução do estudo, os objetivos propostos e as justificativas para a sua realização. Já no capítulo 2 apresenta-se a fundamentação teórica, contendo as definições dos problemas de otimização, supermercados para abastecimento de linhas de montagem, fluxo de peças em linhas de montagem de automóveis, problemas básicos de roteamento nos supermercados e algoritmos baseados em solução única, conteúdo que dará subsídio à construção do modelo a ser proposto para a roteirização de veículos durante o reabastecimento de linhas de produção.

No capítulo 3 é apresentado o método utilizado para o desenvolvimento do estudo e seus objetivos, levantamento bibliográfico dos trabalhos que retratam a aplicação de meta-heurísticas para solução de problemas de roteirização e *scheduling* e detalhes do algoritmo desenvolvido. No capítulo 4 são apresentados os cenários integrantes do modelo de otimização proposto por meio do algoritmo desenvolvido, bem como os cálculos de desempenho do algoritmo baseado no custo computacional agregado e, em paralelo, os resultados obtidos a partir da execução do algoritmo e da comparação de cenários propostos. No capítulo 5 são apresentadas, por fim, as considerações finais deste estudo.

2 FUNDAMENTAÇÃO TEÓRICA

A fundamentação teórica proposta para a problemática de roteirização de *tow trains* para reabastecimento de linhas de montagem automotiva, conta, em primeiro plano, com a apresentação da teoria dos problemas de otimização, a fim de contextualizar a natureza matemática da definição de rotas em linhas de montagem mistas. Após tal, explicita-se a definição de supermercados e o fluxo de materiais em uma montadora, dando ênfase no escoamento entre supermercado e estações de trabalho.

Em sequência, é feita uma revisão do problema básico de roteamento, cujo equacionamento será explorado. Ao final, é apresentada a teoria quanto às meta-heurísticas utilizadas para obter respostas de maneira iterativa.

2.1 PROBLEMAS DE OTIMIZAÇÃO

Problemas de otimização são aqueles em que se tem como objetivo a maximização ou minimização de uma função, chamada de função objetivo. Durante o processo de obtenção da solução ótima, as restrições condicionais impostas às variáveis de decisão são responsáveis por determinar a abordagem necessária para configurar a natureza da problemática, bem como as técnicas de programação mais adequadas para resolvê-la (BROGIATO, 2021).

Jourdan *et al.*(2009), afirmam que problemas de otimização podem ser divididos entre algoritmos exatos (AE) e algoritmos aproximados (AA). Os AE são caracterizados por, em um tempo computacional razoável, retornar soluções ótimas para a função objetivo. Já os AA, ainda que retornem uma resposta adequada para a função, não necessariamente fornecem a solução otimizada.

Aponta-se que, de acordo com a estrutura do algoritmo desenvolvido e o esforço computacional necessário para obter uma solução adequada (ainda que não otimizada), os problemas de otimização podem ser classificados como P ou NP-Hard. Os problemas pertencentes à classe P são aqueles que podem ser resolvidos, sem um alto custo computacional, por meio de algoritmos exatos. Em contrapartida, a classe NP-Hard é integrada por problemáticas complexas que tendem a retornar melhores soluções ao serem resolvidas com o auxílio de algoritmos aproximados e técnicas iterativas (FERNANDES, 2016).

Levando-se em conta a diferenciação na qualidade das respostas obtidas de acordo com a categoria na qual o problema de otimização se encontra, é necessário definir os conceitos de soluções ótimas globais (SOG) e soluções ótimas locais (SOL) (MULLER, 2004). Os ótimos locais são caracterizados por apresentarem o maior (ou

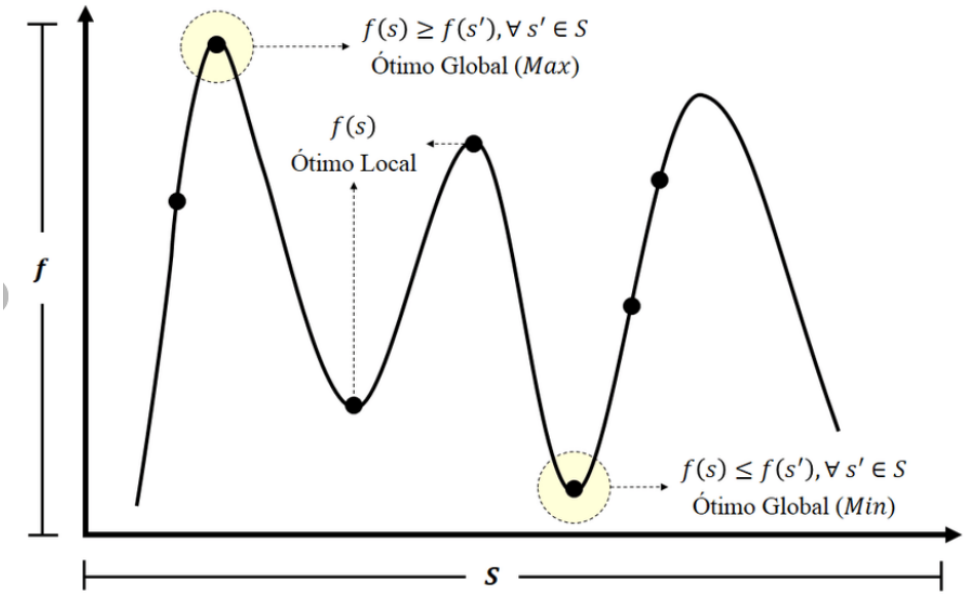
menor de acordo com a proposição do equacionamento) valor da vizinhança, enquanto os ótimos globais maximizam (ou minimizam) a função objetivo (MULLER, 2004).

Equaciona-se essa situação, desse modo, de acordo com a Equação 1.

$$f(X^*) < f(X) \ \forall \ X \ \in \ S \tag{1}$$

Se 1 for válida para todo domínio, X* é um ótimo global, contudo se ela for válida apenas para uma vizinhança N próxima a X*, diz-se que X* é um ótimo local. A Figura 2 exemplifica o conceito de ótimo local e global.

Figura 2 – Solução ótima local e global



Fonte: Adaptado de Fernandes (2019, p. 23).

Na Figura 2 demonstra-se o conceito de que a vizinhança $N(x)$ de X é a quantidade de soluções encontradas no espaço de busca a uma distância ε com $\varepsilon > 0$. Assim, um ótimo local será aquele que representa uma solução $X \ \epsilon \ S$, com S representando o espaço de busca, que apresenta melhor qualidade que todos os vizinhos $S' \ \epsilon \ N(X)$. Já o ótimo global será representado por uma solução $X \ \epsilon \ S$ que apresenta melhor qualidade que todas as outras soluções em todo o espaço de busca $S \leq S'$, $\forall \ S' \ \epsilon \ S$.

No que tange aos problemas de roteirização em supermercados de abastecimento de linhas de montagem, é necessário utilizar-se de métodos solucionais que busquem evitar ótimos locais, a fim de obter a solução mais eficiente para o problema proposto. Para tal, em trabalhos que intentam otimizar o roteamento nestes supermercados, sugere-se o desenvolvimento de algoritmos que utilizem meta-

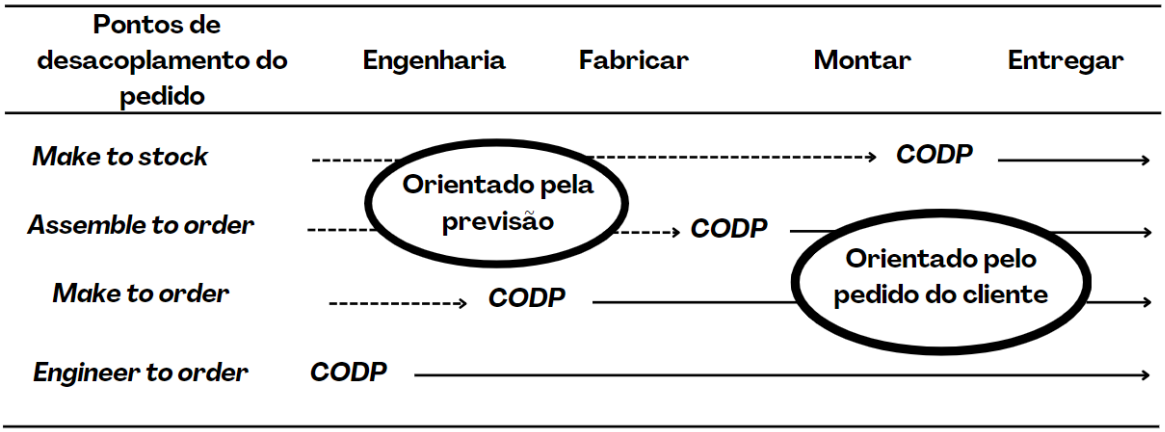
heurísticas baseadas em métodos de busca (CUNHA; WU, 2008).

Tais métodos são caracterizados pela adoção de formulações que não englobam todas as soluções possíveis do problema, e sim, tentam encontrar os melhores caminhos, ou os mais viáveis em tempo computacional razoável, através da avaliação baseada nas características do problema (CARNEIRO, 2010). Desse modo, compreende-se que as soluções, ainda que não ótimas, tendem a apresentar uma qualidade adequada para o problema definido.

2.2 SUPERMERCADOS PARA ABASTECIMENTO DE LINHAS DE MONTAGEM

Ari e Hoang (1995) dividem os ambientes produtivos em quatro categorias: Make to Stock (MTS), Assemble to Order (ATO), Make to Order (MTO) e Engineer to Order (ETO). A estratégia ATO baseia-se na alocação de peças básicas e subconjuntos na montadora e posterior montagem dos produtos finais após fornecidas as informações de demanda dos clientes. Divide-se, assim, a fabricação geral nas etapas: de aquisição ou produção dos subconjuntos e montagem (BATTINI *et al.*, 2013). A Figura 3 demonstra os pontos de desacoplamento do pedido do cliente (CODP) para cada um desses ambientes de produção.

Figura 3 – Ambientes produtivos



Fonte: Adaptado de Olhager (2011, p. 38).

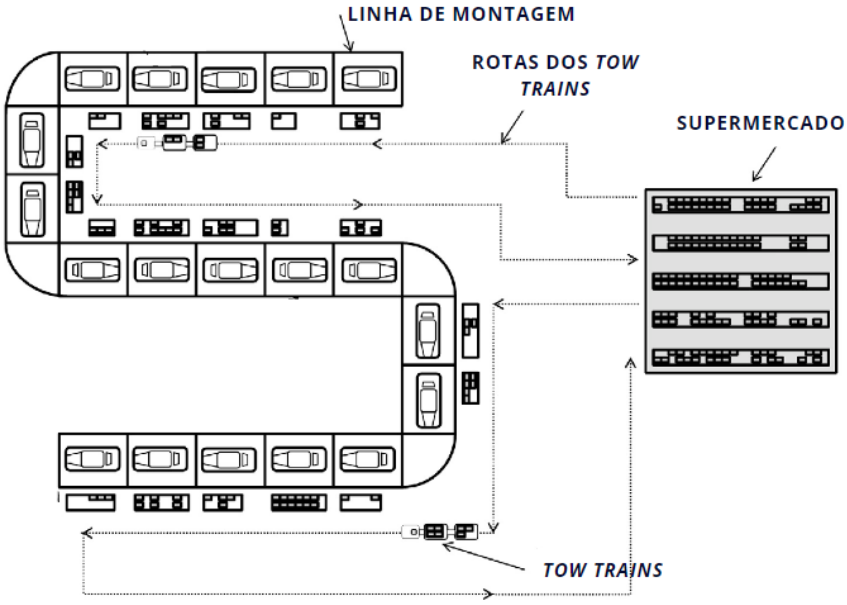
De acordo com Olhager (2011) o desacoplamento ocorre quando o pedido do cliente desvincula-se do restante do fluxo produtivo para definir qual parte será produzida para estoque e qual parte do fluxo será produzida por encomenda. Com isso, demonstra-se a relevância da escolha do ponto de desacoplamento correto para a obtenção de uma resposta rápida à demanda para as montadoras se manterem competitivas no mercado.

Para poder satisfazer todas as exigências dos clientes, as empresas no setor

automotivo têm de oferecer uma ampla gama de diferentes produtos (FACCIO, *et al.*, 2013). Tais montadoras caracterizaram-se pela aderência ao sistema de produção Just in time e aos ambientes produtivos Assemble to Order, criando a necessidade de formalizar o processo de logística interna das fábricas, parte do processo logístico geral que se concentra apenas no gerenciamento de materiais e fluxo de informações dentro da empresa (CAUX; DURIEUX; SAAIDIA, 2014).

Nesse contexto foi criado o conceito dos supermercados de abastecimento de linhas de montagem, sendo estes uma área de armazenamento descentralizada a qual atua como depósito intermediário para as peças necessárias na linha de montagem (ALI, 2019). Possibilita-se, desse modo, a alocação temporária nos supermercados, das peças que serão utilizadas para montar os produtos finais, a fim de reduzir a distância de deslocamento dos *tow trains* e o tempo de transporte das áreas centrais de recebimento às linhas de produção (FACCIO, *et al.*, 2013) A Figura **??** demonstra a disposição do supermercado e das linhas de montagem em uma montadora de automóveis.

Figura 4 – *Layout* de uma montadora de automóveis



Fonte: Adaptado de Emde e Gendreau (2017, p. 2).

São armazenadas nos supermercados diferentes peças e subconjuntos, criando a necessidade de utilizar equipamentos especiais de armazenamento, como prateleiras para carregamento gravitacional ou tubos modulares (ALI, 2019). Em algumas montadoras, os *trolleys* e caixas utilizados para estocar e mover peças do supermercado para as linhas de montagem são feitos sob medida para o tipo de peça, a fim de garantir um transporte seguro e minimizar desperdícios na linha (FACCIO et al.,

2013). A Figura 5 exemplifica o comboio de *trolleys* (veículo controlado pelo operador) atrelados aos *tow trains* (carrinhos ligados em sequência ao veículo rebocador), como comumente realizados em montadoras de automóveis (CUNHA; WU, 2008).

Figura 5 – Acoplamento de trolleys ao *tow train*

Fonte: Adaptado de Emde, Diefenbach e Glock (2020, p. 3).

O uso das prateleiras, que são inclinadas e contém rolamentos para auxiliar no deslocamento das peças pela ação da gravidade (Figura 7), apresenta vantagens ergonômicas frente a prateleiras horizontais comuns, pois, ao permitir aos operadores alcançarem as caixas o mais próximo possível de si, possibilita-se o reabastecimento de diversas peças por vez e reduz-se a necessidade de mover as caixas manualmente (HALIM, *et al.*, 2012).

Figura 6 – Prateleiras gravitacionais

Fonte: Elaboração própria (2023).

Em virtude de tais diferenças construtivas dos equipamentos de alocação, o fluxo de materiais do supermercado até a linha de montagem, tende a depender do tipo de caixa e *trolleys* utilizados para armazenar peças.

2.3 FLUXO DE PEÇAS EM UMA LINHA DE MONTAGEM DE AUTOMÓVEIS

O processo de abastecimento das linhas de montagem de automóveis inicia no recebimento das peças em docas de carga e descarga e, na posterior seleção daquelas que serão destinadas aos supermercados e as que serão alocadas nos armazéns. Em sequência, as peças que foram selecionadas para serem estocadas nos supermercados são embaladas em caixas especiais, de acordo com os padrões específicos de cada componente, e pré-selecionadas de maneira que possibilitem transporte e acesso fluídos (ALI, 2019).

Para tal, rotula-se cada caixa com um código de barras único e essas são destinadas a compartimentos e setores exclusivos para o tipo de peça. Isso ajuda no manuseio de peças dentro do supermercado e permite fácil acesso durante a seleção pelo operador das peças necessárias para posterior transporte à linha de montagem (CUNHA; WU, 2008).

Com isso, forma-se o supermercado a partir de estantes, divididas em seções com prateleiras definidas por localizações específicas para cada peça ou submontagem. Assim, cada tipo de peça tem apenas um local padronizado para todo o supermercado, mesmo que a alocação se altere em período de tempo (ALI, 2019). A Figura 7 mostra um exemplo de como as peças são armazenadas em uma prateleira de supermercado

(ALI, 2019).

Figura 7 – Alocação no supermercado

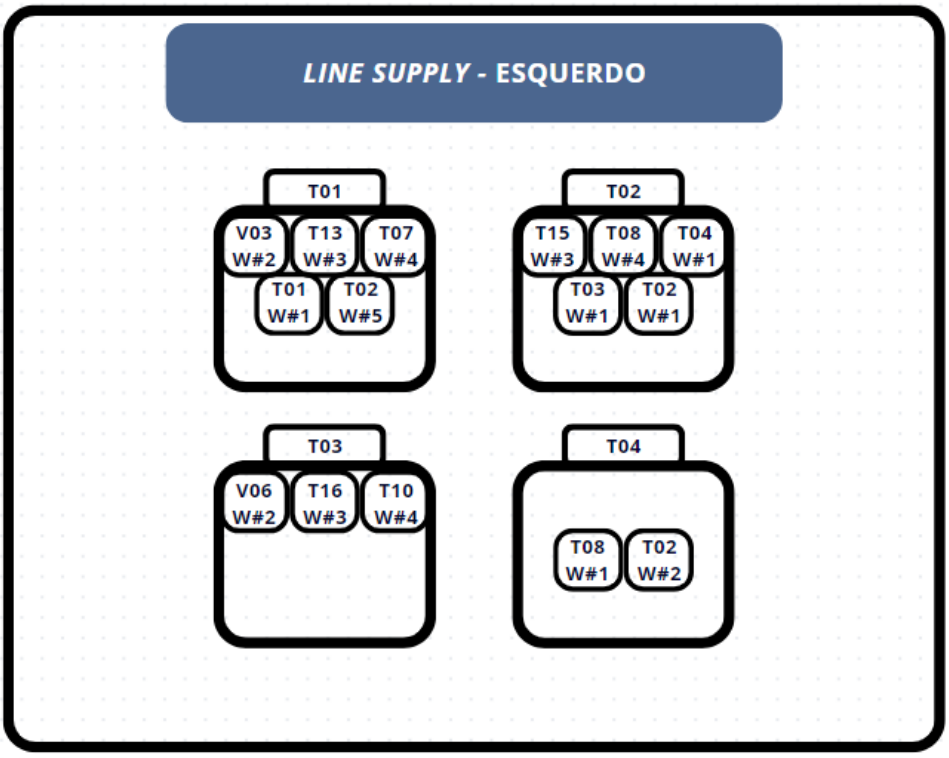
Fonte: Adaptado de Ali (2019, p. 7).

Nessa, visualiza-se a alocação única das peças à medida que os códigos de local (L1, L2, L3 ...) indicam as posições específicas para armazenagem das peças (111, 115, 119 ...), respectivamente.

O processo de localização das peças em estoque irá consistir, desse modo, no rastreio do posicionamento de cada peça no supermercado, por meio dos códigos de barras e das setorizações pré-definidas (ALI, 2019). Durante a etapa, as *Picking Lists*, (listas contendo as ordens de peças a serem coletadas para reabastecer a linha de montagem) e as correspondentes *Waves* (ondas em que ocorrerão as entregas de peças na linha de motagem), irão informar o número do *rack* do supermercado do qual as peças serão retiradas, a correspondente seção, prateleiras, caixa, número da peça, quantidade de peças entregue e hora da entrega (EMDE; POLTEN, 2019). No decorrer da coleta e entrega das peças, os operadores logísticos escaneiam os códigos de barras para manter o registro de quando e onde as peças são retiradas e onde devem ser entregues (ALI, 2019).

A Figura 8 exemplifica a rotina de *waves* para abastecimento de peças em uma linha de montagem (*line supply*) de um montadora. Pode-se verificar que a mesma apresenta o local inicial de retirada das caixas pelos *tow trains* (representado pelos blocos superiores contendo os setores "Ts"), a identificação da caixa a ser retirada (indicada pelo número superior contido nos retângulos menores) e a sua respectiva Wave (representada pelo símbolo "w").

Figura 8 – *Line Supply*



Fonte: Elaboração própria (2023).

Nessa Figura 8, por exemplo, identifica-se que os *trolleys* V03; T13; T07; T01 e T02 (códigos que indicam o destino das peças a serem transportados nos *trolleys*) serão buscados no local T01 do *line supply* esquerdo. Além disso, sabe-se que as *waves*, ou seja, o número de vezes que essas peças estão sendo entregues na linha de montagem, das peças transportadas nesses *trolleys* são, respectivamente, 2, 3, 4, 1 e 5.

Em seguida, transportam-se as peças solicitadas do supermercado até as estações de trabalho ao longo da linha de montagem por meio de *tow trains*, exemplificados na Figura 9, com capacidade para até 5 *trolleys* (CUNHA; WU, 2008). Esses, também, recolhem os *trolleys* vazios a fim de levá-los para o setor de reabastecimento (supermercado)(CUNHA; WU, 2008). Dessa forma, os supermercados podem entregar lotes de peças frequentes, mas pequenos, minimizando os estoques e as distâncias percorridas.

Figura 9 – *Tow trains*

Fonte: Adaptado de Cunha e Wu (2008, p. 847).

Para organizar essa operação é necessário controle da produção com o intuito de que as peças sejam repostas na linha de montagem antes de acabarem, evitando a parada das linhas. Nas montadoras de automóveis, comumente atreladas com a estratégia de produção ATO, o sistema de controle organizacional mais utilizado é o Kanban. Este é um método visual de controle de materiais, que busca gerenciar a operação de acordo com uma sinalização que transmite a necessidade imediata de reabastecimento, movendo-se ao longo do processo (CUNHA; WU, 2008).

Desse modo sempre que as peças são esgotadas, o Kanban emite um sinal ao repositor de peças indicando a necessidade de reposição para uso posterior, logo os trabalhadores logísticos repõem as peças com a quantidade especificada pelos cartões Kanban e o sistema volta a funcionar de maneira eficiente (ALI, 2019).

O Quadro 1 demonstra algumas das regras de aplicação do Kanban.

Quadro 1 – Kanban

Função do Kanban	Regra aplicada
Infos sobre transporte	Processo após pegar o número de itens indicado pelo Kanban do processo anterior
Infos produção	O processo anterior de de bens de acordo com o número e a sequência mostrado Kanban
Impedir vantagens	Nenhuma mercadoria é transportada sem Kanban
Ordens de serviço	Sempre anexe o Kanban às mercadorias
Prevenir defeito	Produtos defeituosos não são encaminhados para o próximo processo.
Divulgar problemas	O Kanban se ajusta às flutuações na demanda

Fonte: Adaptado de Beber (2009, p. 54).

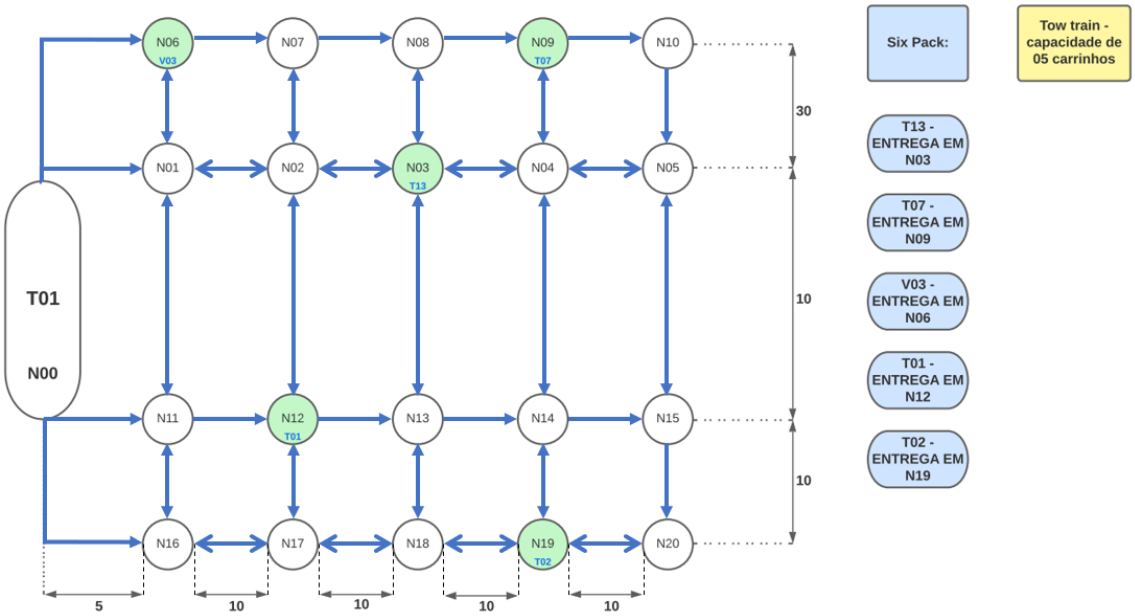
Ainda que os supermercados sejam organizados de acordo com as posições específicas das peças, esses locais são flexíveis e podem ser atualizados sempre que necessário. No ambiente *Assemble-to-Order*, caracterizado pelo dinamismo, a mudança nas demandas do cliente faz com que alguns desses locais mudem após cada período de tempo, geralmente mensalmente (ALI, 2019).

Cria-se, desse modo, a necessidade de atualizar os sistemas de roteirização próprios de cada montadora de automóveis (planilhas Excel, roteirizações manuais, sistemas de roteirização) com os novos posicionamentos do supermercado e fornecer novos relatórios (contendo as novas rotas ótimas) aos operadores logísticos, a fim de definir novas rotas de reabastecimento das linhas de montagens, foco de estudo deste trabalho. A Figura 4, anteriormente exposta, exemplifica um cenário de definição de rotas em uma montadora.

Indica-se, que nessa Figura 4 as ruas que poderão ser utilizadas na geração de rotas dos *tow trains*, bem como a localização do supermercado e as estações de trabalho na linha de montagem são destacados. Neste trabalho, para a definição das rotas a serem percorridas no reabastecimento das linhas de montagem será realizada a análise dos melhores caminhos a serem percorridos, de modo que se minimize a distância e o tempo de transporte.

Aponta-se, ainda, que a fim de simplificar a visualização de fluxos de peças entre o supermercado e as estações dispostas ao longo da linha de montagem, é possível esquematizar a estrutura da montadora em termos de fluxos (Figura 10).

Figura 10 – Esquema de roteamento

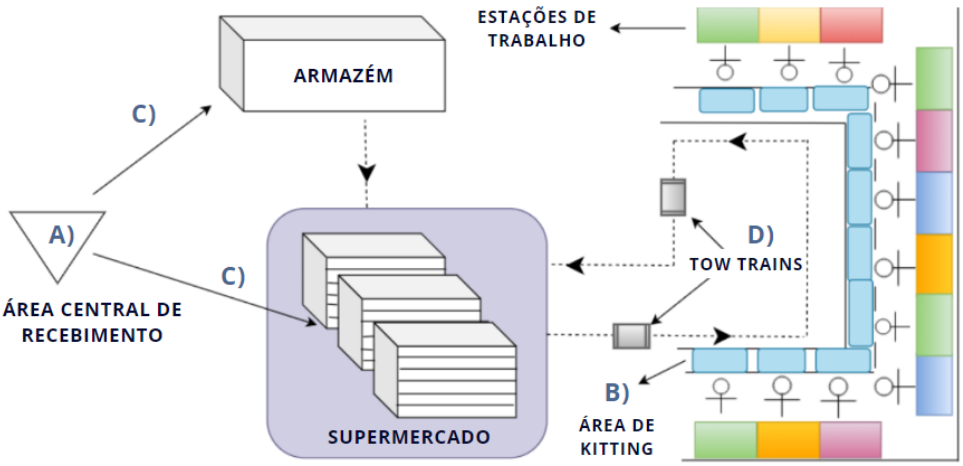


Fonte: Elaboração própria (2023).

Na Figura 10, os círculos representam as estações de trabalho (N01, N02, ... N20), as setas azuis as direções permitidas para a circulação dos *tow trains*, as setas pretas a distância entre cada estação de trabalho e a elipse (T01) representa o supermercado. Um cenário de demanda para reabastecimento pode ser exemplificado pela solicitação de peças nas estações N03, N06, N09, N12 e N16 (destacadas em verde); a solicitação de peças é transmitida, então, aos operadores de *tow trains* que se destinam ao supermercado (T01) a fim de recolher os *trolleys* que contém as peças solicitadas. Na sequência, os operadores devem percorrer a rota a ser definida, respeitando os sentidos delimitados, e entregar os *trolleys* em cada estação correspondente; por fim, os operadores retornam ao supermercado para reabastecer os *tow trains* e atenderem uma nova solicitação de peças.

Desse modo, destaca-se que o processo de abastecimento de linhas de montagem automotiva é dividido nas seguintes etapas: a) recebimento das peças nas docas de carga/descarga, b) seleção das peças que serão destinadas aos supermercados e aos armazéns, c) preparação, em *trolleys*, das peças estocadas nos supermercados, para posterior envio às linhas de montagem, e d) realização do transporte (rotas) para entrega de peças na linha (ALI, 2019). Tal preparação subdivide-se, ainda, na alocação das peças do supermercado em caixas especiais, de acordo com os padrões específicos de cada componente, e a pré-seleção dessas, de maneira que o transporte e acesso sejam possibilitados como evidenciado na Figura 11 (EMDE; POLTEN, 2019).

Figura 11 – Processo de abastecimento



Fonte: Elaboração própria (2023).

Neste estudo de caso, as *picking lists* representam a ferramenta adotada a ser utilizada a fim de localizar as peças (inicialmente em estoque e posteriormente em trânsito) por meio dos códigos de barras inseridos nestas e das setorizações pré-definidas nos supermercados (localizações direcionadas para alocação de determinados produtos). Essas são entregues aos operadores dos *tow trains* e aos funcionários responsáveis por carregarem os *trolleys* com as peças indicadas, e conterão informações relativas aos pedidos de peças, as correspondentes *waves*, o número dos *trolleys* por pedido, seções em que as peças estão localizadas no supermercado, quantidade a ser entregue e momento a ser entregue na linha de montagem.

Desse modo, torna-se possível organizar e planejar as movimentações de peças ao longo da linha a partir dessas listas, bem como, dividir a escala de operação entre os operadores dos *tow trains* (CUNHA; WU, 2008).Uma forma de melhor gerir o abastecimento de forma eficiente é através do sistema de gestão e controle Kanban. As vantagens da adoção desse sistema, apoiam-se na verificação contínua das atividades e processos exercidos na linha, por meio de reuniões diárias com os funcionários, além de atualizações de quadro e monitores (FACCIO et al., 2013).

Utilizando-se de uma prática comum em montadoras automotivas, pode-se dividir o supermercado em dois fluxos principais, representados pelo i) *body shop* e ii) pelos armazéns, de acordo com o padrão de escoamento de peças adotado nesses. Em i) se armazenam as peças usadas com mais frequência, enquanto em ii) se armazenam as peças usadas com menos frequência, de modo que o processo de alocação e transporte dessas sejam facilitados (ALI, 2019).

Quanto à alocação dos *trolleys* e peças especiais, aponta-se que essa ocorre

de maneira a facilitar o processo de locomoção dos *tow trains*, de modo que necessita-se de planejamento prévio para sua definição (FERNANDES, 2016). No ambiente ATO de uma montadora automotiva, a mudança nas demandas do cliente resulta na atualização periódica, geralmente mensal, desses fluxos, criando a necessidade de replanejar o *layout* geral dos supermercados (FERNANDES, 2016).

2.4 PROBLEMA BÁSICO DE ROTEAMENTO NO SUPERMERCADO

O problema de roteirização em supermercados é definido como uma integração do problema de roteirização de veículos e da gestão de estoques, ou seja, define-se a quem, quando e quanto entregar de peças, levando em consideração a rota a ser percorrida (ALI, 2019). De modo semelhante Fernandes (2016) define que este problema destaca-se como um dos mais complexos na gestão de operações, ao passo que integra a gestão de estoque e as decisões de roteirização de veículos, intentando minimizar os custos logísticos globais e satisfazer as demandas dos clientes.

Ainda, Diaz (2020) aponta que um problema de roteirização em supermercados, caracteriza-se pela determinação simultânea do plano de reabastecimento das estações dispostas na linha de montagem ao longo de um horizonte de planejamento e do plano de roteirização de veículos em cada período de funcionamento da linha. Desse modo, em virtude da integração entre os problemas de roteirização de veículos e da gestão de estoques, comprova-se a necessidade de analisar a roteirização em supermercados por meio de um *trade-off* entre custos de transporte/roteamento e custos de manutenção de estoque, sendo essa uma abordagem mais realista quando comparado com soluções que tratam de ambos os problemas separadamente (FERNANDES, 2016).

A modelagem matemática que representa o funcionamento de uma linha de montagem mista de uma montadora de automóveis é constituída conjuntamente do problema de roteirização e programação de *tow trains*, os quais devem ser adaptados a cada turno de atuação da montadora. O problema de roteirização caracteriza-se pela tentativa de determinar o tamanho da frota de *tow trains* a ser utilizada durante o reabastecimento de peças na linha de montagem, bem como, a minimização da distância total percorrida por esses (EMDE; BOYSEN, 2011).

Além disso, a definição da sequência de escalas mostra-se relevante à medida que revela-se necessário atender às estações da linha de montagem em tempos específicos a fim de evitar atrasos na operação. Infere-se, que um facilitador dessa programação de operações surge devido ao acoplamento de *trolleys* e o raio de giro relativamente grande em comparação com os corredores de uma linha de montagem, o que impossibilita a manobra de retorno no corredor de origem (EMDE; BOYSEN, 2011). Assim, os passeios restringem-se a unidirecionais ao longo da direção do fluxo da linha de montagem.

Outro ponto relevante na modelagem do problema mostra-se na manutenção de uma extensa frota de *tow trains*, a qual envolve custos de aquisição, manutenção e pessoal altos, mostrando-se desejável manter esse número mínimo (CUNHA; WU, 2008). Com isso, o problema de *schedulling* deve ser resolvido para cada *tow train* e seu correspondente conjunto de estações a serem atendidas em cada *wave*, destacando-se a série de objetivos operacionais visados para reduzir os custos da operação (Tabela 1).

Tabela 1 – Objetivos operacionais

Objetivo	**Reflexos/Justificativa**	**Tipo**
Reduzir a distância total percorrida	Redução de custos variáveis relativos à utilização de equipamentos	Secundário
Reduzir frota de *tow trains*	Redução de custos de aquisição, manutenção e pessoal	Prioritário

Fonte: Elaboração própria (2023).

De acordo com Cunha e Wu (2008), o tempo de viagem de um *tow train* pode ser dividido em quatro parcelas:

- Tempo de preparação da carga (parar e descer do rebocador; identificar itens a serem transportados; montar o comboio; subir no *tow train* e partir.
- Tempo de deslocamento, o qual pode ser estimado como o quociente entre a distância total da rota e a velocidade do rebocador.
- Tempo de entrega das peças na linha de montagem (parar e descer do rebocador; transferir as peças para a linha; subir no *tow train* e partir.
- Reduções de velocidade ou paradas.

Ainda assim, destaca-se que, em virtude das curtas distâncias entre o supermercado e as estações e a breve duração das visitas, as rotas aqui retratada podem ser aproximadas, tendo durações idênticas. Desse modo, o problema em questão reduz-se à determinação do número de rotas e a janela de tempo de realização de cada uma dessas.

A fim de determinar tais variáveis, mostra-se necessário analisar a demanda de peças a serem entregues nas linhas de montagem, a qual é determinada pela sequência de produção nas mesmas. Como há a necessidade de realizar a programação dos *tow trains* diariamente e em cada turno de produção, a sequência de produção deve ser previamente conhecida e as demandas de peças para cada estação e ciclo de produção definidas.

Portanto, algumas restrições operacionais que devem ser consideradas a fim de garantir o abastecimento correto e seguro da linha (Tabela 2).

Tabela 2 – Restrições operacionais

Restrição	Justificativa	Tipo
Número máximo de *trolleys* rebocados	Restrições físicas para a movimentação, capacidade de tração do rebocador e segurança	Obrigatória
Rotas fixas por rebocador	Atribuição de responsabilidade ao operador	Obrigatória
Atendimento de estações localizados do lado esquerdo da linha	Evitar *tow trains* na contramão; segurança; evitar congestionamentos.	Obrigatória
Peças selecionadas em uma mesma rota	Movimentação do rebocador em regiões concentradas da área de produção.	Desejada
Rebocadores não fazem manobra de retorno no corredor de onde vieram	Inviabilidade técnica, segurança.	Obrigatória

Fonte: Adaptado de Cunha e Wu (2008, p. 850).

Desse modo, sendo um problema NP-Hard, como definido na Seção 2.1, e muitas vezes apresentando instâncias de grande porte ou não tão facilmente estruturadas, a roteirização de *tow trains* em linhas de montagem tende a implementar soluções de roteirização recorrendo a métodos heurísticos, principalmente meta-heurísticas, para obter soluções viáveis (FERNANDES, 2016). De acordo com Jourdan, Basseur e Talbi (2009), a fim de utilizar-se meta-heurísticas para resolução de um problema de otimização é válido checar alguns fatores como:

- A complexidade do problema;
- O tamanho das instâncias, uma vez que mesmo em problemas *NP-hard*, se esses possuírem instâncias pequenas, podem ser viáveis para serem resolvidos com algoritmos exatos;
- O tempo pretendido de busca (alguns métodos focam em pesquisar rapidamente para poder entregar a melhor solução possível em um intervalo de tempo relativamente curto, e outros em garantir a otimização e desconsiderar a importância de reduzir o tempo computacional).

Com isso, levando-se em consideração que o problema de roteirização e *scheduling* de *tow trains* em linhas de montagem pertencem à classe *NP-hard* e, ao empregar inúmeras condicionais e restrições, tendem a não poder ser reduzidos em problemas menores (redução de instâncias), comprova-se a validade da aplicação de meta-heurísticas para sua solução.

Cordeau et al. (2007) descrevem as principais meta-heurísticas propostas para a solução de tal problema destacando três classes principais:

- As meta-heurísticas baseadas em métodos de busca, como *simulated annealing*,

deterministic annealing e *tabu search*

- Meta-heurísticas baseadas em métodos populacionais, como o algoritmo genético.
- *Learning mechanisms*, como redes neurais.

Já Fuchigami (2005) classifica os métodos heurísticos em dois grupos distintos: métodos construtivos, a partir da ordenação das tarefas segundo índices de prioridade; métodos melhorativos, nos quais obtém-se uma solução inicial e, em seguida, através de algum procedimento iterativo, procura-se uma melhor programação das tarefas que a atual, de acordo com a medida de desempenho proposta. No que tange aos algoritmos utilizados para otimização de rotas por meio de um processo iterativo, os métodos melhorativos de heurística destacam-se.

Outra referência na abordagem da solução de problemas de roteirização por meio de heurísticas é o artigo produzido por Hasle e Kloster (2007), o qual foca na heurística *rich vehicle routing problems* para a solução de problemas de roteirização ligados à indústria (coleta e abastecimento). De acordo com os autores, os problemas de roteirização tratados na indústria possuem caráter complexo e, a menos que possuam aplicações específicas ou cenários reduzidos de simulação, só podem ser resolvidos satisfatoriamente por meio de heurísticas.

Aponta-se que Cunha e Wu (2008) abordaram o problema de roteirização em supermercados por meio da proposição de uma heurística construtiva inspirada na heurística de Solomon (1987). O estudo desenvolvido possuiu como objetivo roteirizar uma frota de *tow trains* para o transporte de embalagens (do supermercado até a linha de montagem), buscando minimizar o número de veículos utilizados e a distância total percorrida. A fim de atingir tal objetivo, a heurística atua na construção de rotas e, em sequência, na seleção de estações ainda não roteirizadas para a inserção na rota corrente até que novas alocações não possam ser realizadas (em virtude de restrições de capacidade ou devido ao tempo total de viagens).

Emde e Boysen (2011) abordam o problema de reabastecimento de peças aplicado em uma linha de montagem de automóveis. A abordagem adotada compreende a utilização de programação dinâmica para a resolução simultânea de problemas de roteirização e *schedule* dos *tow trains*, de modo que o cenário de uma montadora seja satisfatoriamente reproduzido. Nesse cenário, o objetivo é determinar as rotas e os cronogramas ótimos, bem como os *trade-offs* entre número de viagens e a capacidade de transporte dos *tow trains*.

Golz et al. (2012), de maneira semelhante, empregam como alvo de estudo o reabastecimento de linhas de montagem de automóveis sob o sistema *Just in Time*, adotando como ponto central a região do super mercado. Para a resolução do problema, utilizam-se uma solução heurística que visa minimizar o número necessário de motoristas para viagens internas entre o supermercado e os locais de entrega designados. Ainda, Vaidyanathan *et al.* (1999) estudam o problema de roteirização para

o reabastecimento de peças em uma fábrica trabalhando sob o sistema *Just in Time*, por meio de uma heurística de duas fases. A fim de simplificar a formulação, algumas considerações são tomadas, como a adoção de uma demanda por peças constante ao longo do tempo e do abastecimento por uma frota homogênea de *tow trains* a partir do supermercado.

No que tange a utilização de meta-heurísticas para a resolução do problema de roteirização em linhas de montagem de supermercados, o estudo de Fernandes (2016) destaca-se por empregar dois métodos distintos: o *simulated annealing* e o *Greedy Randomized Adaptive Search Procedure (GRASP)*. Durante a elaboração do modelo matemático a ser empregado para obter as soluções de roteirização, o autor cria dois cenários a partir das meta-heurísticas distintas a fim de comparar o método mais eficiente, chegando a conclusão de que os dois métodos apresentam resultados semelhantes.

De maneira análoga, Ali (2019) emprega a programação dinâmica a fim de otimizar o fluxo de material em linhas de montagem dos supermercados de montadoras de automóveis e, portanto, reduzir o custo devido ao movimento de realocação das linhas. O método proposto no estudo foi avaliado no contexto da fábrica de montagem de motores de automóveis Volvo em Skövde.

Diaz (2020), emprega duas meta-heurísticas para realizar o problema de roteirização de *tow trains* em linhas de montagem, sendo essas a *simulated annealing* e a *local search*. O estudo comprovou, também, que as duas meta-heurísticas apresentaram resultados semelhantes para pequenas, médias e grandes instâncias. A Tabela 3 apresenta o resumo desses estudos, destacando os autores, a natureza base do problema a ser resolvido e o método de solução escolhido.

Tabela 3 – Resumo dos estudos destacados

Autor	Ano	Problema base	Método de solução
Diaz	2020	Roteirização de *tow trains* em linhas de montagem	*Simulated annealing* e *local search*
Ali	2019	Otimização do fluxo de material em linhas de montagem dos supermercados de montadoras de automóveis	Programação dinâmica
Fernandes	2016	Roteirização em linhas de montagem de supermercados	*Simulated annealing* e GRASP
Golz	2012	Reabastecimento de linhas de montagem de automóveis sob o sistema *Just in Time*	Solução heurística que visa minimizar o número de motoristas para viagens internas entre o supermercado e os locais de entrega
Emde e Boysen	2011	Problema de reabastecimento de peças aplicado em uma linha de montagem de automóveis	Utilização de programação dinâmica
Cunha e Wu	2008	Possuiu como objetivo roteirizar uma frota de *tow trains* para o transporte de embalagens buscando minimizar o número de veículos utilizados e a distância total percorrida	Heurística construtiva inspirada na heurística de Solomon (1987).
Cordeau *et al.*	2007	Principais meta-heurísticas para a solução de problemas de roteirização	Meta-heurísticas baseadas em métodos de busca; Meta-heurísticas baseadas em métodos populacionais; *Learning Mechanisms*
Hasle e Kloster	2007	Solução de problemas de roteirização ligados à indústria (coleta e abastecimento)	Heurística *rich vehicle routing problems*
Fuchigami	2005	Classificação de meta-heurísticas em dois grupos: construtivos e melhorativos	Os métodos melhorativos de heurística destacam-se
Vaidyanathan *et al.*	1999	Problema de roteirização para o reabastecimento de peças em uma fábrica trabalhando sob o sistema *Just in Time*	Heurística de duas fases

Fonte: Elaboração própria (2023).

2.5 ALGORITMOS BASEADOS EM SOLUÇÃO ÚNICA

Algoritmos baseados em solução única são aqueles que geram uma solução genérica inicial e, ao compará-la com a vizinhança, buscam melhorias que podem substituir a solução inicial, em um processo iterativo, até que um critério de parada definido anteriormente seja satisfeito (ALI, 2019). Ao comparar-se as meta-heurísticas com as heurísticas clássicas, percebe-se que essas tendem a ampliar o espaço de pesquisa das soluções do problema e evitar ótimos locais (CUNHA; WU, 2008).

Infere-se que, dependendo do tamanho da vizinhança e da qualidade da solução inicial no espaço de busca, as meta-heurísticas podem apresentar alto custo computacional, em virtude do alto número de nós e da dificuldade de encontrar um ótimo global a partir de uma solução inicial não adequada, fato que reduz a confiabilidade da solução obtida. A partir da definição de uma vizinhança reduzida, os algoritmos tornam-se mais eficazes por contemplarem todas as soluções possíveis no espaço em um tempo computacional adequado (JOURDAN *et al.*, 2009).

Com o intuito de evitar a parada do algoritmo em um ótimo local devido a qualidade da solução inicial, há a possibilidade de utilizar-se métodos perturbadores para mudar a vizinhança a fim de encontrar um novo ótimo local melhorado. Nesse sentido, as meta-heurísticas de buscas locais, as quais consistem em cruzar o espaço de busca realizando iterativamente modificações na solução inicial tentando encontrar um ótimo, destacam-se pela eficiência e qualidade das soluções (ALI, 2019).

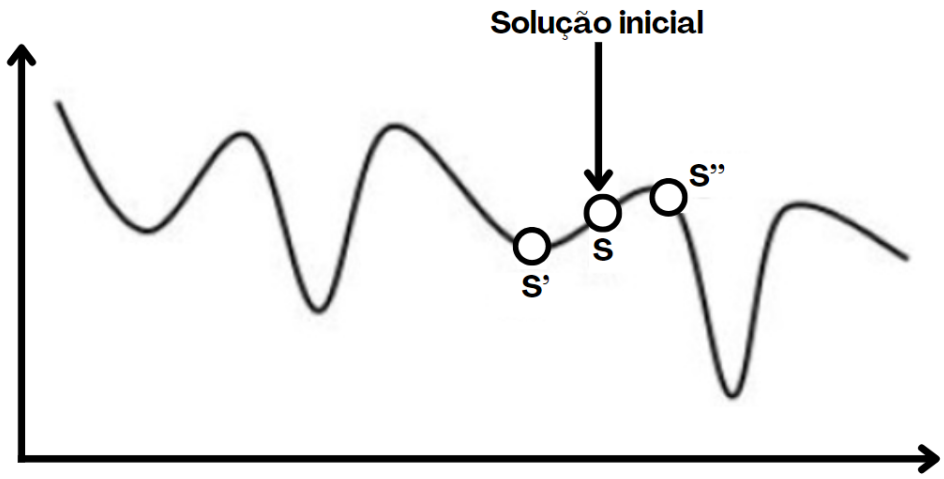
Entre tais meta-heurísticas pode-se destacar as técnicas de *simulated annealing* (SA), busca tabu (*Tabu Search*), *greedy randomized adaptive search procedure* (GRASP), *Guided Local Search* (GLS) e *variable neighborhood search* (VNS) (CUNHA; WU, 2008). Durante o desenvolvimento de algoritmos focados na melhoria de rotas, a técnica de *simulated annealing*, um algoritmo de busca local proposto por (KIRKPATRICK; GELATT; VECCHI, 1983), possui destaque, em virtude do foco na solução de problemas de otimização global a partir da analogia com o processo de recozimento de metais, no qual a obtenção de uma estrutura cristalina resistente está associada ao aquecimento da substância seguido de um resfriamento lento (SANTOS, 2016).

De acordo com Ribeiro (2018), a analogia entre o Simulated Annealing (SA) e o processo de recozimento de metais ocorre de maneira que os diversos estados que um metal pode assumir durante o recozimento são comparados às soluções disponíveis no espaço de busca do problema de otimização, a energia associada a cada estado metalúrgico encontra um paralelo na função objetivo que se busca otimizar e, por fim, a energia mínima atingida durante o recozimento reflete uma solução ótima, seja ela local ou global, no contexto do problema em questão. Com isso, proporciona-se uma compreensão visual e intuitiva do funcionamento do SA, reforçando a ideia de busca por estados de menor energia, ou seja, soluções ótimas, à medida que o algoritmo avança em seu processo de otimização.

Essa heurística de busca local é empregada em problemas de otimização a fim de potencializar a busca de soluções em regiões mais próximas que possibilitem a fuga de ótimos locais (FERNANDES, 2016). A grande vantagem do SA é que, intentando escapar de mínimos locais, esse algoritmo permite a aceitação de soluções de piora, ou seja, torna-se possível admitir soluções piores para que se possa alcançar outras regiões dentro do espaço de vizinhança e assim chegar a uma solução mais próxima

do ótimo global (GOMES, 2003). A Figura 12 permite a observação desta definição, pois, mostra que partindo de uma solução $f(S)$, é possível encontrar a solução $f(S')$ em sua vizinhança, de tal forma que forma que, $f(S') < f(S)$ ou $f(S'') > f(S)$.

Figura 12 – Vizinhança de busca



Fonte: Elaboração própria (2023).

Ribeiro (2018), destaca, ainda, que o Simulated Annealing inicia seu processo de otimização a partir de uma solução inicial, que pode ser obtida aleatoriamente ou por meio de algum algoritmo determinístico. A cada iteração, um novo estado é gerado por meio de uma modificação aleatória do estado atual. Em seguida, se o novo estado possui uma energia menor que o estado atual, ele é adotado como o estado presente. No entanto, caso o novo estado tenha uma energia maior que o estado atual, o SA utiliza um método probabilístico para determinar se esse novo estado será aceito ou não. Essa abordagem, que incorpora aleatoriedade e um critério probabilístico, confere ao SA a flexibilidade necessária para explorar diferentes regiões do espaço de busca e escapar de mínimos locais, contribuindo para sua eficácia na busca por soluções ótimas.

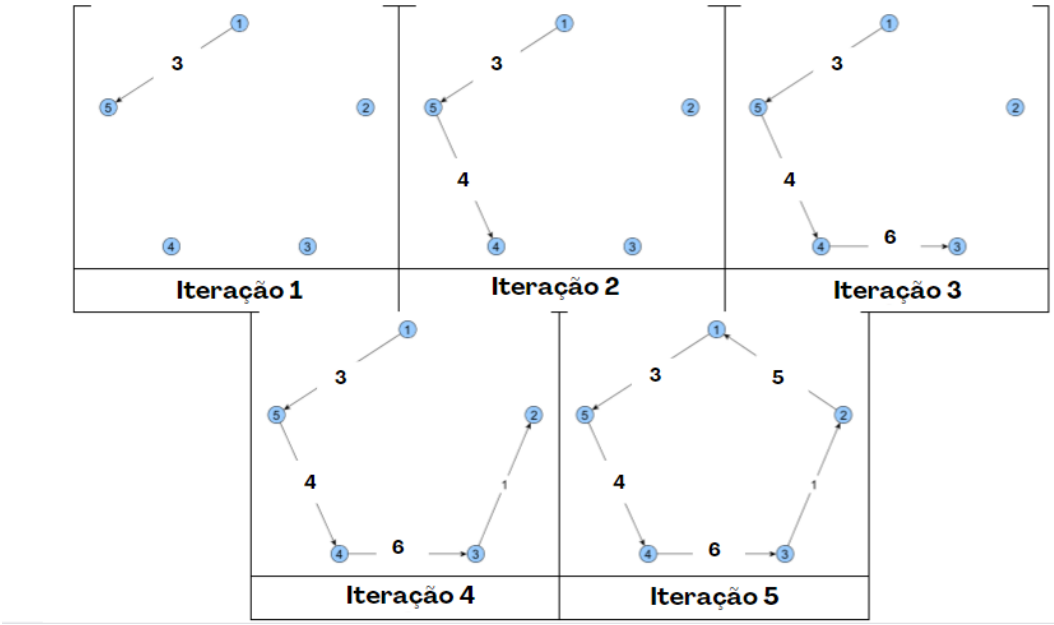
Definidas tais vantagens, adotou-se na pesquisa o *Simulated Annealing* como técnica principal de busca no algoritmo de otimização das rotas dos *tow trains*, ainda que esse apresente um custo computacional razoavelmente elevado em função do espaço de busca extenso comparado com o método de vizinhos mais próximos (ARAUJO, 2001).

Ainda, a fim de realizar uma comparação de custos computacionais, optou-se por empregar um método de busca distinto no algoritmo, o qual tende a adotar estratégias menos custosas para solucionar problemas de roteirização: a heurística do vizinho mais próximo.

De acordo com Bellmore e Nemhauser (1968), esse método caracteriza-se como uma heurística construtiva, portanto, a cada passo um componente da solução é adicionado. Tal heurística baseia-se, tomando como início o vértice inicial de um grafo, em partir a cada passo para um vértice vizinho de menor custo que ainda não esteja inserido na solução, até que a rota esteja estruturada. O processo repete-se de maneira iterativa, sendo que, a cada iteração, o nó inicial é substituído a fim de minimizar o efeito da escolha do nó e, por fim, a solução com menor custo total é escolhida (GOLDBARG; LUN, 2005).

A Figura 13 demonstra um exemplo de aplicação dessa heurística. No grafo, visualiza-se que, ao iniciar-se o roteiro pelo vértice 1, tem-se como solução o trajeto 1; 5; 4; 3; 2; 1 e um custo final de 18 u.m.

Figura 13 – Vizinho Mais próximo



Fonte: Elaboração própria (2023).

Embora a heurística do Vizinho Mais Próximo apresente uma abordagem direta e rápida para a construção de rotas, por se tratar de uma heurística gulosa (constrói a solução por meio de uma sequência de decisões, em que cada decisão visa obter uma solução ótima local), essa apresenta algumas limitações consideráveis. Uma desvantagem liga-se com o fato do método se concentrar apenas no vizinho mais próximo em cada etapa, tendendo a encontrar soluções ótimas locais, uma vez que pode não considerar a melhor opção global para otimização do percurso (ARENALES, 2015). Em contraponto, essa heurística apresenta uma grande vantagem devido a simplicidade de implementação e ao baixo custo computacional, tornando-a um método largamente aplicado em problemas de grande escala em que encontrar a solução exata

é impraticável devido à complexidade computacional (ARENALES, 2015).

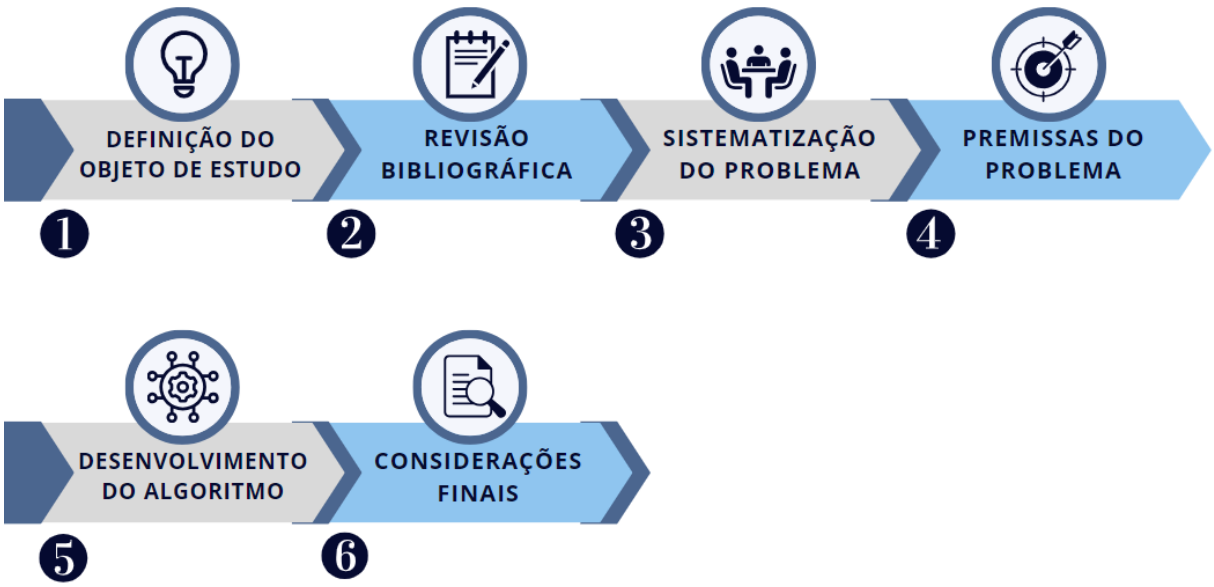
Definidos os métodos de busca a serem aplicados no algoritmo, suas vantagens e desvantagens, permite-se iniciar o processo de concepção e teste do código. Este se dará no capítulo seguinte.

3 MÉTODO

Este capítulo apresenta a descrição dos métodos de pesquisa utilizados, bem como formulações selecionadas para resolução do problema de roteirização de veículos para abastecimento de linhas de produção e etapas sequenciais de implementação computacional em Python dos métodos selecionados.

Também são descritas as premissas adotadas para a construção do modelo de roteirização, a sistematização de análise do custo computacional empregado e o processo da obtenção da solução ótima global, na geração de rotas a serem percorridas pelos *tow trains*. A Figura 14 exibe o fluxograma da execução das etapas deste trabalho.

Figura 14 – Fluxograma das etapas do trabalho



Fonte: Elaboração própria (2023).

3.1 OBJETO DE ESTUDO

O presente estudo propõe o desenvolvimento de um algoritmo para a resolução do problema de roteirização e *scheduling* de *tow trains* em linhas de montagem mistas de montadoras de automóveis. Para tal, definiram-se os objetivos gerais e específicos, apresentados respectivamente na Seção 1.2.

3.2 REVISÃO BIBLIOGRÁFICA

A etapa de levantamento bibliográfico desenvolvida nesse trabalho contemplou os temas de roteirização de *tow trains*, supermercados de montadoras de automóveis, abastecimento de linhas de produção e métodos computacionais de resolução de problemas de *scheduling* e roteirização. Definido o escopo de pesquisa, realizou-se a busca de trabalhos acadêmicos para avaliar as abordagens apresentadas na literatura para o planejamento do reabastecimento de linhas de montagem voltadas ao sistema *Just in Time*. O propósito desta busca foi de encontrar trabalhos que corroborassem o uso de heurísticas e meta-heurísticas integradas com métodos computacionais para a resolução do problema aqui proposto.

Desse modo, utilizou-se as base Scopus com a combinação das palavras-chave inclusas no título, resumo e palavras-chave: *routing AND supermarket AND meta-heuristics AND programming AND optimization AND scheduling*. Visando garantir a credibilidade das fontes utilizadas como referencial para o estudo, a busca foi restringida para artigos e *conference papers* e como resultado foram encontrados 72 documentos.

Após a leitura dos títulos e resumos dos trabalhos pré-selecionados, optou-se por excluir 50 documentos, uma vez que esses tangenciavam o objetivo da busca sem, contudo, estarem diretamente relacionados. Com a leitura dos 22 documentos restantes, percebeu-se que 2 artigos retratavam a utilização de heurísticas e meta-heurística para resolução de problemas de roteirização e *scheduling* de maneira genérica, ou seja, sem aplicação em um cenário real de indústrias. Já 17 artigos aplicaram tais métodos de solução em indústrias diversas, adaptando a otimização de rotas de veículos para solução de problemas como redução de desperdício de alimentos perecíveis ou redução de emissões.

Por fim, 3 artigos aplicam as heurísticas e meta-heurísticas em ambientes de montadoras de automóveis sob o sistema *Just in Time*, objetivando roteirizar a frota de *tow trains* empregadas para o reabastecimento da linha de montagem. Com isso, para os 22 documentos encontrados que se adequavam ao foco da pesquisa, foi realizada uma análise dos problemas focais e dos métodos de solução, os quais são resumidos ao conteúdo presente na Tabela 4.

Tabela 4 – Revisão bibliográfica

Autor	Ano	Problema base	Método de solução
Karels *et al.*.	2023	Problema de roteirização de veículos (efeito de diferentes acordos de serviços entre os clientes e o prestador de serviços logísticos)	AE (Algoritmo exato)
Liu *et al.*	2023	Otimização das rotas de entrega e estratégias de cobrança de veículos elétricos não tripulados	GVNS (uma variante do VNS que substitui o procedimento de busca local no VNS básico pelo VND)
Song e Wu	2023	Otimização do problema de inventário e roteirização para produtos perecíveis	Heurística de duas fases baseadas na *Simulated Annealing*
Truden, Maier e Armbrust	2022	Problema de roteirização de veículos com janelas de tempo	*Mixed-integer linear programming*
Wang *et al.*	2021	A otimização de redes colaborativas de logística de coleta e entrega em vários depósitos com cargas divididas e janelas de tempo	Solução de duas fases e um algoritmo híbrido
Diaz	2020	Roteirização de *tow trains* em linhas de montagem	*Simulated annealing e local search*
Ali	**2019**	**Otimização do fluxo de material em linhas de montagem dos supermercados de montadoras de automóveis**	**Programação dinâmica**
Nair *et al.*	2018	*Scheduling* e roteirização de operações de resgate e entrega de alimentos	Heurística baseada na *Tabu Search*
Fernandes	2016	Roteirização em linhas de montagem de supermercados	*Simulated annealing* e GRASP
Nambirajan *et al.*.	2016	Problema de roteirização e inventário com reabastecimento	*Integer linear programming* e heurística de três fases
Wang, Li e Hu	2015	Roteirização de veículos com janelas de tempo e uma restrição de carregamento incompatível	*Tabu Search e Heurística*
Wen e Eglese	2015	VRP de custo mínimo	Novo algoritmo heurístico
Li *et al.*	2014	Problema de inventário e roteirização com o objetivo de minimizar o tempo de viagem	*Tabu Search*
Mes, Schutten e Rivera	2014	Roteirização para coleta dinâmica de resíduos	Heurística simples de solução do IRP
Golz	**2012**	**Reabastecimento de linhas de montagem de automóveis sob o sistema *Just in Time***	**Solução heurística que visa minimizar o número de motoristas para viagens internas entre o supermercado e os locais de entrega**
Emde e Boysen	**2011**	**Problema de reabastecimento de peças aplicado em uma linha de montagem de automóveis**	**Utilização de programação dinâmica**
Cunha e Wu	2008	Possuiu como objetivo roteirizar uma frota de *tow trains* para o transporte de embalagens buscando minimizar o número de veículos utilizados e a distância total percorrida.	Heurística construtiva inspirada na heurística de Solomon (1987).
Cordeau *et al.*.	2007	Principais meta-heurísticas para a solução de problemas de roteirização	Meta-heurísticas baseadas em métodos de busca; Meta-heurísticas baseadas em métodos populacionais; *Learning Mechanisms*
Hasle e Kloster	2007	Solução de problemas de roteirização ligados à indústria (coleta e abastecimento)	Heurística rich vehicle routing problems
Fuchigami	2005	Classificação das meta-heurísticas em dois grupos: construtivos e melhorativos	Os métodos melhorativos de heurística destacam-se
Vaidyanathan *et al.*	1999	Problema de roteirização para o reabastecimento de peças em uma fábrica trabalhando sob o sistema *Just in Time*	Heurística de duas fases

Fonte: Elaboração própria (2023).

Em virtude da maior aproximação com o tema focal do presente trabalho, aponta-se que os artigos aplicados diretamente nos ambientes de montadora de automóveis serão utilizados como base referencial para o desenvolvimento da metodologia aqui proposta e esses podem ser observados em negrito na Tabela 4.

3.3 SISTEMATIZAÇÃO DO PROBLEMA PROPOSTO

Ao longo deste tópico, objetiva-se descrever as etapas e processos contidos no processo de pré-concepção da modelagem lógica e matemática do problema de roteirização proposto. Desse modo, serão apresentados os fluxogramas que regem a teoria da solução de otimização, bem como a lógica que apoia as expressões matemáticas adotadas para o equacionamento da problemática a ser resolvida.

3.3.1 Formulação lógica

Tradicionalmente, os problemas de roteirização apresentam natureza iterativa, fato que cria a necessidade da adoção de um processo dinâmico para a formulação lógica das ferramentas desenvolvidas para a solução destes casos (EMDE; POLTEN, 2019). Em virtude de tal, inspirando-se na metodologia clássica de projeto, este trabalho de conclusão de curso apoia-se no sequenciamento lógico de desenvolvimento de um algoritmo otimizador baseado na Espiral de Evans. Esta, representa uma sequência lógica de tarefas a serem realizadas iterativamente a fim de solucionar o problema alvo, promove uma convergência mais eficiente para uma solução viável, e cuja construção, pode ser feita utilizando-se técnicas de tomada de decisão tais quais matriz de influência ou matriz de decisão (ROSSI, 2018).

Propõe-se, então, o desenvolvimento de uma matriz de decisão (MD) que rege a escolha das tarefas contidas na espiral de projetos. O resultado de tal subtarefa pode ser observado na Figura 15, sendo que as instruções de preenchimento da matriz contavam com o preenchimento da matriz de decisão linha por linha, assumindo o valor 1 se o processo vertical precisar ser realizado/determinado antes do processo horizontal.

O preenchimento da matriz de decisão auxiliou o processo de definição da ordem de implementação das restrições e condicionais no código, de modo que os parâmetros com maior somatória devem ser implementados de maneira antecedente aos demais. Além disso, a matriz revela os processos que podem ser implementados em paralelo, facilitando o processo de codificação e criação do algoritmo.

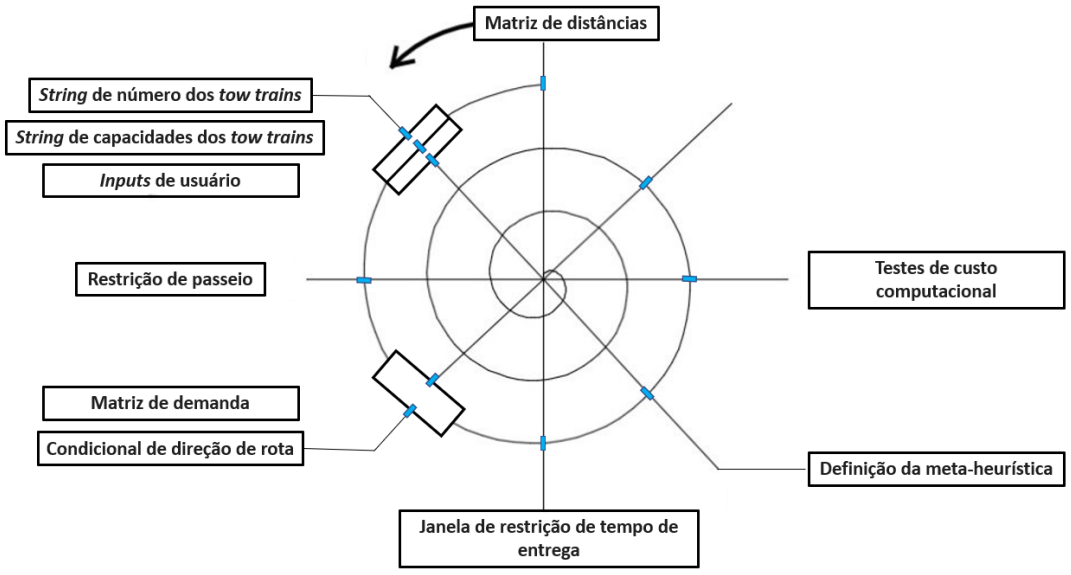
Figura 15 – Matriz de decisão

Matriz de Decisão											
Etapas	Matriz de distâncias	Matriz de demanda	*String* de número dos *tow trains*	*String* de capacidades dos *tow trains*	Condicional de direção de rota	Restrição de passeio	Janela de restrição de tempo de entrega	*Inputs* de usuário	Definição da meta-heurística	Testes de custo computacional	Soma
Matriz de distâncias	x	1	1	1	1	1	1	0	1	1	8
Matriz de demanda	0	x	0	0	1	0	0	0	1	1	3
String de número dos *tow trains*	0	0	x	0	1	1	1	0	1	1	5
String de capacidades dos *tow trains*	0	0	0	x	1	1	1	0	1	1	5
Condicional de direção de rota	1	0	0	0	x	0	0	1	0	1	3
Restrição de passeio	0	0	0	0	1	x	1	0	1	1	4
Janela de restrição de tempo de entrega	0	0	0	0	0	0	x	0	1	1	2
Inputs de usuário	0	0	0	0	1	1	1	x	1	1	5
Definição da meta-heurística	0	0	0	0	0	0	0	0	x	1	1
Testes de custo computacional	0	0	0	0	0	0	0	0	0	x	0

Fonte: Elaboração própria (2023).

Definida a MD e os pesos referentes a cada tarefa, possibilitou-se o desenvolvimento da Espiral de Evans, a qual integrou os requisitos estabelecidos para o funcionamento de uma linha de montagem de uma montadora de automóveis (restrições e condicionais definidas ao problema), os desejos de projeto e o sequenciamento indicado na matriz de decisão. Nota-se que, ao longo da primeira volta da espiral, as tarefas ligam-se exclusivamente aos requisitos estabelecidos, definindo a primeira etapa de projeto de acordo com a Figura 16 (ROSSI, 2018).

Figura 16 – Espiral de Evans



Fonte: Elaboração própria (2023).

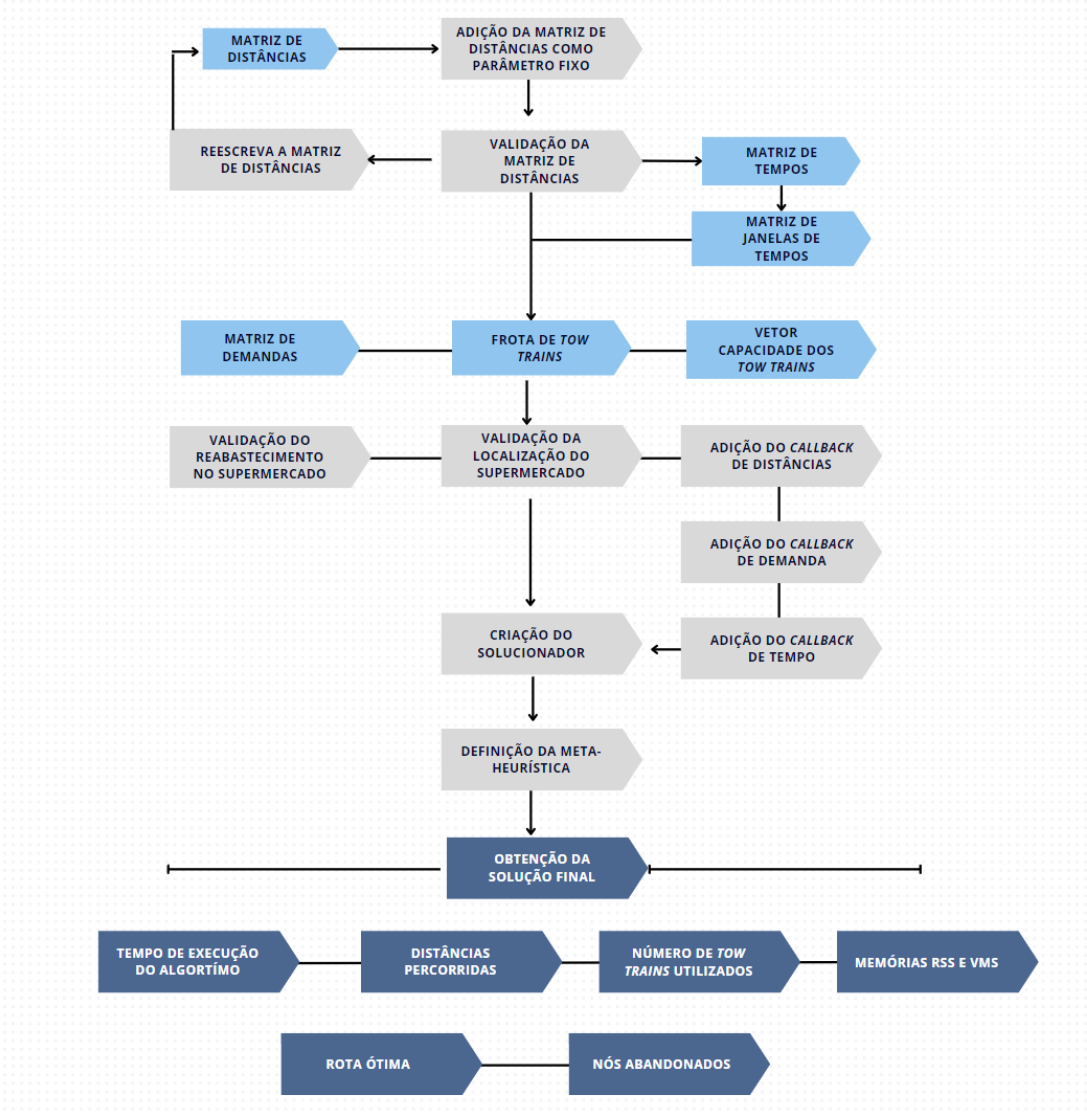
Seguindo com a definição de lógica do algoritmo, após a finalização da primeira volta na espiral ocorre o retorno ao ponto inicial a fim de verificar a viabilidade e eficiência das soluções adotas ou, eventualmente, modificar algumas das características definidas, comprovando o caráter iterativo da ferramenta. A partir da segunda volta, abordam-se as necessidades da montadora de automóveis e a implementação de acordo com a matriz de dificuldades.

Lamb (2003) afirma que a cada volta na espiral, permite-se uma menor gama de variação no projeto, limitando-se a capacidade da exploração de soluções consideravelmente distintas daquelas analisadas nas fases iniciais do projeto. Desse modo, reitera-se a importância do retorno às condições iniciais definidas para o desenvolvimento do algoritmo a fim de garantir que essas são, de fato, as mais eficientes possíveis na obtenção da solução.

Finalizada a Espiral de Evans, desenvolveu-se um fluxograma (Figura 17) com as etapas especificas de desenvolvimento da ferramenta computacional heurística de planejamento de rotas de *tow trains* para reabastecimento de peças em linhas de montagem automotiva, consideradas neste trabalho.

O fluxograma, representado na Figura 17, desempenha um papel crucial ao fornecer uma visão visual e estruturada das etapas específicas envolvidas no desenvolvimento do algoritmo, apresentando as diferentes fases do algoritmo, destacando *inputs*, *outputs* e os passos fundamentais de implementação. Nas fases iniciais, os principais *inputs* são identificados, abrangendo informações essenciais sobre a infraestrutura da linha de montagem, demandas de abastecimento e restrições logísticas. Os passos subsequentes do algoritmo são apresentados de maneira sequencial e lógica, delineando o processo de tomada de decisão que orienta o planejamento do código. Os *outputs*, em sequência, definidos, abrangendo as rotas otimizadas, as distâncias percorridas, o número de *tow trains* necessários para realizar o abastecimento da linha, e outros indicadores de desempenho do algoritmo para realização de testes de custo computacional.

Figura 17 – Fluxograma de desenvolvimento do algoritmo



Fonte: Elaboração própria (2023).

Aponta-se que, nesse fluxograma, os *inputs* do algoritmo estão destacados nos blocos em azul claro. Já os blocos de cor azul escuro representam a solução e os *outputs* gerados. Por fim, os blocos em cinza representam os passos de implementação lógica do código criado.

3.3.2 Estratégias e simplificações adotadas

Definidas as condições a serem aplicadas, a fim de propor uma ferramenta computacional heurística de apoio ao reabastecimento de linhas de montagem (Seção 2.4), foram consideradas as seguintes estratégias e simplificações:

- As unidades de tempo são padronizadas para percursos equidistantes.
- Para evitar congestionamentos nos corredores estreitos, as estações na rota de

um rebocador são sempre atendidas na direção do fluxo da linha de montagem.
- Todas as peças transportadas são de tamanho padronizado. Com uma capacidade máxima de *trolleys* por *tow trains* e caixas padronizadas, a restrição de capacidade dos veículos pode ser medida unidimensionalmente, limitando o número K de caixas a serem carregadas. Como todos os rebocadores possuem capacidade de carregar a carga proposta, uma restrição de peso adicional não é um problema.
- Todos os *tow trains* partem com a capacidade máxima de *trolleys* atrelados
- Depois de partir, um *tow train* sempre percorrerá todas as estações em sua rota sem interrupção. A duração de cada parada não depende da carga, de modo que todas as rotas de um único *tow train* têem a mesma duração.
- A sequência de produção é previamente determinada, de modo que a demanda em cada estação é conhecida e imutável.

3.4 PREMISSAS DO PROBLEMA

A fim de averiguar o cenário de estudo para desenvolvimento da ferramenta computacional meta-heurística, torna-se viável analisar as condicionais e restrições apresentadas no problema e os inputs e outputs esperados, em virtude da aplicação dessas.

3.4.1 Condicionais e restrições

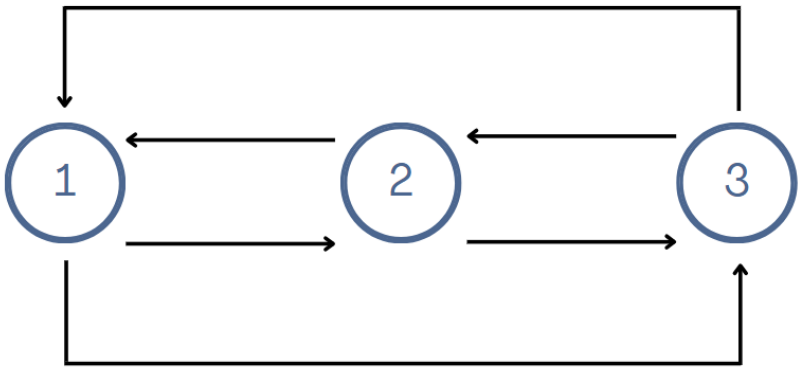
No problema de roteirização de veículos aqui retratado, define-se como objetivo principal encontrar as melhores rotas para diversos veículos que visitam um conjunto de locais e retornam, por fim, ao local inicial da rota. Conceitua-se, portanto, as melhores rotas como aquelas que apresentam a menor distância total de percurso e atendem todas as condições propostas pelo operador.

A fim de implementar as visitas aos nós distribuídos ao longo da linha de montagem, é válido garantir a existência de um único caminho fechado, de modo que, exista um arco entrando e saindo de cada vértice, ou seja, assegurando um passeio na malha de pontos (KRUK, 2018). Definida a existência dessa condicional, aplica-se uma restrição a fim de garantir, assim que concluídas todas as paradas propostas pelo operador, uma rota única pelos pontos escolhidos.

Um exemplo dessa condicional pode ser observado ao considerar três nós, como na Figura 18. Há nessa, a possibilidade de escolher uma rota simples representada pelo *loop* 1-2-3 (a qual poderia representar a saída do *tow train* do supermercado no nó 1, o atendimento das estações 2 e 3), contudo, caso não seja adicionada uma restrição, não há impedimentos para a realização contínua desse *loop* ou de rotas extras internas como a (1,2,3,2,1,2,1), revelando-se a necessidade de eliminá-los.

Para tal, desconsiderando-se a condicional de retorno ao supermercado para reabastecimento, a restrição adotada é a de que para qualquer subconjunto dos nós o número de arcos escolhidos deve ser menor que o número de nós (KRUK, 2018).

Figura 18 – Exemplo de loop em uma trajetória



Fonte: Elaboração própria (2023).

No caso apresentado na Figura 18, tal restrição adotada para eliminar as rotas extras pode ser verificada na Equação 2.

$$(x_{1,2} + x_{2,1} + x_{2,3} + x_{3,2} + x_{1,3} + x_{3,1}) < 3 \tag{2}$$

Considerando a necessidade de incorporar a condicional de retorno ao supermercado para reabastecimento, mantém-se a restrição para assegurar o passeio único pela malha de pontos, porém adiciona-se a rota que liga o último nó visitado ao primeiro nó do passeio. Assim, para o exemplo da Figura 2, a rota resultante após a adição da condicional de retorno ao supermercado seria definida pelo *loop* 1-2-3-1.

Além disso, levando-se em conta que cada *tow train* possui uma capacidade de transporte limitada e esses precisam retirar os *trolleys* no supermercado para levá- los até a linha de montagem, é necessário considerar-se tal capacidade máxima (comboio de 5 *trolleys*) de carregamento para otimizar os trajetos percorridos por cada veículo. Dessa forma, é preciso adicionar uma nova restrição ao problema.

A restrição será apontada por meio do limite de operação do *tow train* empregado comumente em montadoras automotivas, equivalente ao transporte de 5 *trolleys* em comboio. Deve- se assumir, paralelamente, que cada *tow train* parte do supermercado totalmente carregado, descarrega apenas um *trolley* por nó (estação de trabalho na linha de produção) e é recarregado completamente após voltar ao ponto inicial (supermercado).

Além disso, para essa restrição funcionar, é necessário inserir um dado referente à demanda em cada um dos nós distribuídos na malha representativa da

planta fabril. Assim, deve-se buscar a rota que possua a distância percorrida total mais curta, que atenda a demanda de cada nó e, ainda assim, nunca ultrapasse a capacidade de transporte de cada *tow train* (EMDE; BOYSEN, 2011).

No que tange aos *trolleys*, uma restrição de demanda deve ser aplicada de maneira semelhante ao realizado com os *tow train*. Ao considerar a demanda associada com cada estação de trabalho, garante-se que os *trolleys* não excedam sua capacidade máxima de carga ao atender todas as demandas das estações visitadas (KRUK, 2018).

Com isso, a soma das demandas em uma rota não devem exceder a capacidade máxima dos *trolleys* transportados. Desse modo, mostra-se necessário selecionar as estações a serem abastecidas de tal forma que, ao considerar suas demandas, a capacidade dos *trolleys* utilizados na rota não seja ultrapassada.

Outra restrição a ser considerada trata das janelas de tempo para se realizar a entrega das peças na linha de montagem, visando atendê-las sem causar paradas ou gargalos; portanto, é necessário realizar a entrega das peças nas estações de trabalho em períodos específicos. Para tal, utiliza-se o conceito de matrizes de tempos que deverão conter o tempo total de percurso entre cada um dos nós (PERRON; FURNON, 2023).

O último grupo de restrições trata das direções das ruas integrantes das *line supplies*, visto que em uma fábrica pode-se definir que algumas vias podem ser bidirecionadas e outras percorridas em apenas uma direção. A restrição adotada, deste modo, deverá regular a ida e vinda dos *tow trains* nos *loops* restritos pelas direções.

3.4.2 *Inputs* e *outputs* do modelo proposto

Para inicializar a modelagem da ferramenta computacional meta-heurística para roteirização de veículos e abastecimento de linhas de produção deve-se definir quais são os dados necessários para inserção no código computacional, à medida que objetiva-se simular de maneira verossímil o cenário fabril de uma montadora de veículos. Infere-se que esses serão divididos entre dados fixos (adicionados na raiz do código em virtude da sua não alteração no contexto da montadora) ou dados fornecidos pelo operador (dados requisitados pelo algoritmo ao usuário da ferramenta).

Entre os *inputs* aqui descritos, alguns dados foram espelhados dos trabalhos descritos na etapa de referencial bibliográfico e, adicionalmente, foram realizadas visitas técnicas a montadoras de automóveis a fim de verificar a validade dos dados obtidos e sua aplicabilidade a cenários reais, incluindo conversas e reuniões com gestores e profissionais da área para melhor entender cada parâmetro adotado.

Vale ressaltar que, embora os *inputs* do algoritmo aqui desenvolvido não tenham sido extraídos de uma única montadora de automóveis, eles representam cenários semelhantes aos encontrados em fábricas reais. Isso garante uma base sólida e aplicável para as análises e conclusões deste estudo.

Assim, o primeiro *input* indicado é uma matriz de distâncias, como exemplificado na Figura 19, composta pelas informações de posicionamento referentes a cada nó na planta da montadora de automóveis (supermercado ou estações de trabalho ao longo da linha de montagem) e das distâncias entre estes. Essa será inserida como um parâmetro fixo, no qual as distâncias serão pré-calculadas a fim de evitar a computação no momento de execução do algoritmo, otimizando-o (KRUK, 2018).

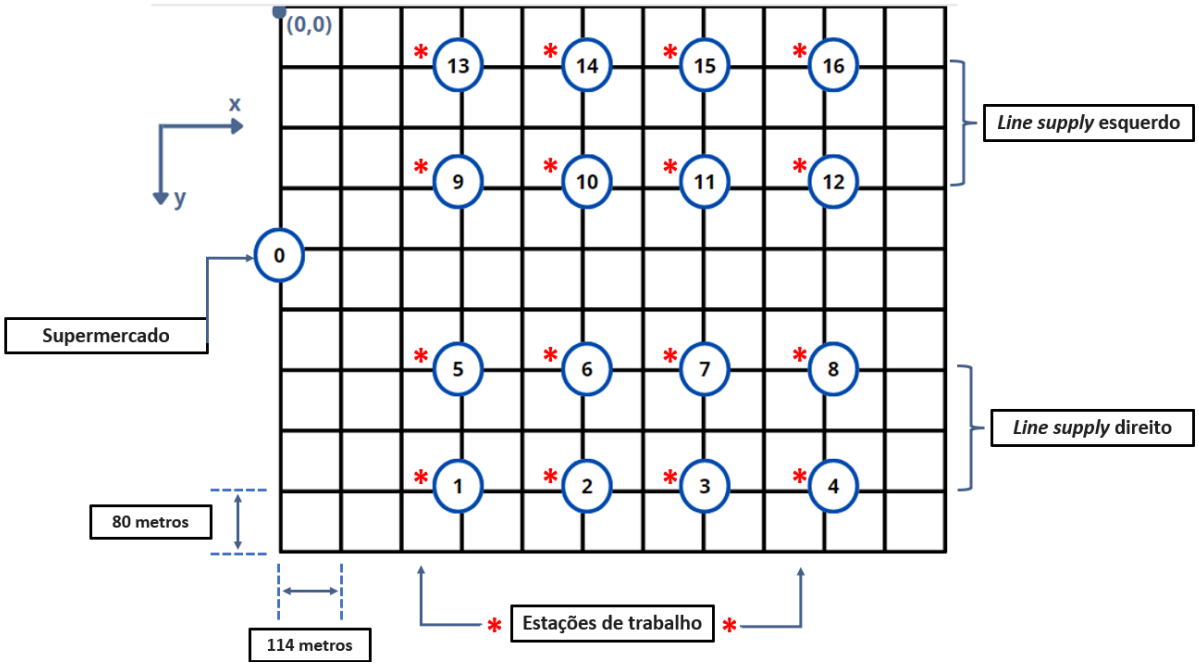
Figura 19 – Exemplo de matriz de distâncias genérica considerando 10 nós

P (x y)	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9
P0 72 19		711	107	516	387	408	539	309	566	771
P1 10 37	539		769	881	380	546	655	443	295	1140
P2 77 31	122	752		281	441	264	318	448	588	730
P3 89 61	519	875	274		435	334	93	776	949	302
P4 51 61	484	561	338	419		118	268	607	495	431
P5 57 52	409	406	244	380	93		295	544	549	494
P6 82 69	479	735	334	101	345	247		679	809	238
P7 52 1	221	444	433	744	487	435	649		325	840
P8 21 14	510	303	599	984	531	553	847	350		1001
P9 88 96	663	989	664	335	588	434	297	1093	1012	

Fonte: Elaboração própria (2023).

No que tange à matriz de distâncias criada para a inserção no algoritmo, definiu-se que esta contaria com 16 estações de trabalho (numeradas de 1 à 16) distribuídas entre 4 linhas de montagem (4 estações por linha), estrutura de distribuição comum no *layout* de montadoras de automóveis divididas em *line supply* esquerdo e direito. Para obter essa matriz nos moldes da matriz de exemplo apresentada na Figura 19 atribuíram-se coordenadas $x - y$ aos nós representativos das estações, como apresentado no diagrama (Figura 20).

Figura 20 – Diagrama de distribuição das estações na linha de montagem



Fonte: Elaboração própria (2023).

Nesse, apresenta-se uma exibição visual do posicionamento de cada estação de trabalho de uma linha de montagem automotiva, possibilitando a etapa de pré-computação da matriz de distâncias. Destaca-se que, a fim de fornecer as coordenadas $x - y$ para os nós indicados no diagrama, o presente trabalho baseou-se nas médias de valores de distâncias tipicamente apresentados em montadores de automóveis sob o sistema JIT.

Com isso, após definidas as distâncias relativas à planta apresentada na Figura 20, onde a distância vertical entre cada nó equivale a 80 metros e a distância horizontal equivale a 114 metros, o posicionamento do supermercado no nó de número 0, a distribuição de estações de trabalho e a posição da origem nas coordenadas da planta, pode-se estimar as seguintes coordenadas para cada um dos pontos:

- (0, 320) - localização 0 (supermercado)
- (342, 640) - localização 1
- (570, 640) - localização 2
- (798, 640) - localização 3
- (1.026, 640) - localização 4
- (342, 480) - localização 5
- (570, 480) - localização 6
- (798, 480) - localização 7
- (1.026, 480) - localização 8
- (342, 240) - localização 9

- (570, 240) - localização 10
- (798, 240) - localização 11
- (1.026, 240) - localização 12
- (342, 80) - localização 13
- (570, 80) - localização 14
- (798, 80) - localização 15
- (1.026, 80) - localização 16

Desse modo, ao calcular a distância entre cada uma das estações, elabora-se a matriz de distâncias. Esse cálculo é realizado definindo-se que a distância entre dois pontos $(x1, y1)$ e $(x2, y2)$ será calculada como a distância euclidiana (Equação 3), gerando a matriz 17x17 apresentada na Figura 21.

$$\sqrt{(x1 - x2)^2 + (y1 - y2)^2} \tag{3}$$

Figura 21 – Input matriz de distância

Nós	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0,00	468,36	653,68	859,77	1.074,74	377,58	592,03	813,88	1.038,40	351,23	575,59	802,00	1.029,11	417,81	618,47	833,31	1.053,70
1	468,36	0,00	228,00	456,00	684,00	160,00	278,54	483,26	702,46	400,00	460,42	606,58	792,37	560,00	604,64	722,17	884,00
2	653,68	228,00	0,00	228,00	456,00	278,54	160,00	278,54	483,26	460,42	400,00	460,42	606,58	604,64	560,00	604,64	722,17
3	859,77	456,00	228,00	0,00	228,00	483,26	278,54	160,00	278,54	606,58	460,42	400,00	460,42	722,17	604,64	560,00	604,64
4	1.074,74	684,00	456,00	228,00	0,00	702,46	483,26	278,54	160,00	792,37	606,58	460,42	400,00	884,00	722,17	604,64	560,00
5	377,58	160,00	278,54	483,26	702,46	0,00	228,00	456,00	684,00	240,00	331,03	515,30	724,88	400,00	460,42	606,58	792,37
6	592,03	278,54	160,00	278,54	483,26	228,00	0,00	228,00	456,00	331,03	240,00	331,03	515,30	460,42	400,00	460,42	606,58
7	813,88	483,26	278,54	160,00	278,54	456,00	228,00	0,00	228,00	515,30	331,03	240,00	331,03	606,58	460,42	400,00	460,42
8	1.038,40	702,46	483,26	278,54	160,00	684,00	456,00	228,00	0,00	724,88	515,30	331,03	240,00	792,37	606,58	460,42	400,00
9	351,23	400,00	460,42	606,58	792,37	240,00	331,03	515,30	724,88	0,00	228,00	456,00	684,00	160,00	278,54	483,26	702,46
10	575,59	460,42	400,00	460,42	606,58	331,03	240,00	331,03	515,30	228,00	0,00	228,00	456,00	278,54	160,00	278,54	483,26
11	802,00	606,58	460,42	400,00	460,42	515,30	331,03	240,00	331,03	456,00	228,00	0,00	228,00	483,26	278,54	160,00	278,54
12	1.029,11	792,37	606,58	460,42	400,00	724,88	515,30	331,03	240,00	684,00	456,00	228,00	0,00	702,46	483,26	278,54	160,00
13	417,81	560,00	604,64	722,17	884,00	400,00	460,42	606,58	792,37	160,00	278,54	483,26	702,46	0,00	228,00	456,00	684,00
14	618,47	604,64	560,00	604,64	722,17	460,42	400,00	460,42	606,58	278,54	160,00	278,54	483,26	228,00	0,00	228,00	456,00
15	833,31	722,17	604,64	560,00	604,64	606,58	460,42	400,00	460,42	483,26	278,54	160,00	278,54	456,00	228,00	0,00	228,00
16	1.053,70	884,00	722,17	604,64	560,00	792,37	606,58	460,42	400,00	702,46	483,26	278,54	160,00	684,00	456,00	228,00	0,00

Fonte: Elaboração própria (2023).

No algoritmo, são considerados como *inputs* fixos o número máximo de *tow trains* utilizados na fábrica, o índice do local representado pelo supermercado na matriz de distâncias, a quantidade máxima de *trolleys* empregados no comboio de um único *tow train*, a capacidade máxima de peças transportadas em um *trolley* e a direção das rotas possíveis entre os pontos delimitadores de uma rota. Esses são responsáveis por delimitar o processo de entrega de peças à medida que aplicam as restrições de capacidade, rota e demanda, mencionadas na Seção 3.4.1.

Desse modo, consideram-se os seguintes valores para cada um dos parâmetros de *inputs* fixos:

- Número máximo de *tow trains* = 5 veículos;

A frota de *tow trains* definiu-se de acordo com uma média de valores verificados durante visitas técnicas à montadoras de automóveis.

- Índice do local representado pelo supermercado na matriz de distâncias = 0;

Definição que impacta a lógica de retorno para reabastecimento dos *tow trains* utilizados para distribuir as peças ao longo das estações na linha de montagem.

- Quantidade máxima de *trolleys* empregados no comboio de um único *tow train* = 5 unidades;

Valor definido em função da média de carregamento máximo apresentada pelos *tow trains* abordados nos estudos do referencial bibliográfico e das visitas técnicas. Infere-se que a restrição de *trolleys* agregados dá-se em virtude da capacidade de manobra dos *tow trains* e potência máxima de seus motores.

- Capacidade máxima de peças transportadas em um *trolley* = 1;

Valor em função da interpretação do algoritmo da capacidade unitária do *trolley* nas linhas de montagem. Assim, o número de peças torna-se irrelevante à medida que a demanda atende-se com o recebimento de um *trolley*.

- Direção das rotas = mão única entre estações na mesma linha.

Direções utilizadas para simular a capacidade reduzida de manobra dos *tow trains* nas ruas estreitas do *line supply*.

Já os *inputs* fornecidos pelo usuário da ferramenta compreendem: o número de *trolleys* a serem transportados em cada pedido (indicado pelas *Picking Lists*), os nós de retirada (posição do super mercado) e entrega dos *trolleys* (posição das estações na linha de montagem), a matriz de demanda que indica quantas são as peças esperadas em cada estação de trabalho e a matriz de janela de tempo que indica o horário específico em que as peças precisam ser entregues em cada nó (estação de trabalho na linha).

Quanto ao número de *trolleys* a serem transportados em cada pedido, admite-se prioridade para o transporte equivalente a 5 unidades para o *input* do algoritmo, a fim de maximizar, nas situações possíveis, o transporte de peças em cada rota. Já em relação aos nós de retirada e entrega dos *trolleys*, esses podem ser representados por uma matriz na qual o índice 0 representa a retirada e o reabastecimento (no supermercado) e os demais índices representam as posições das estações nas quais as entregas serão realizadas (Figura 22). Para o cenário inicial a ser simulado, opta-se por adicionar uma demanda unitária a todas as estações da linha, objetivando-se verificar as melhores rotas para o atendimento geral da linha de montagem.

Figura 22 – Matriz entrega e retirada

```
# Demandas correspondem apenas aos nós visitados com capacidade unitária

data['demands'] = [1 if i in visited_nodes else 0 for i in range(len(data['distance_matrix']))]

 # Verifique se a demanda está correta
data['demands'] =  [0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

Fonte: Elaboração própria (2023).

No que tange a matriz de demanda, essa é a responsável por indicar quantos são os *trolleys* esperados em cada estação de trabalho e, no presente trabalho ela é representada por: [0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], igualando-se à matriz de entrega e retiradas. Como definido anteriormente, os *trolleys* são tratados no algoritmo como uma unidade, ou seja, as demandas das estações de trabalho não correspondem a um número específico de peças, e sim, a um determinado número de *trolleys*. Desse modo, a matriz de demanda do cenário 0 indica que cada estação da linha de montagem solicitou a visita de 1 *trolley*.

Por fim, a matriz de janela de tempo indica os períodos em que as estações devem ser visitadas a partir do momento em que inicia-se o processo de roteirização, ou seja, cada linha da matriz indica o instante inicial em que um *tow train* pode chegar à estação de trabalho e o último instante em que essa visita é permitida, considerando-se o tempo de viagens entre estações.

As janelas de tempo incluem o intervalo de visitas em minutos, podendo ser indicada por (Figura 23). Essa foi definida a partir de valores médios de janelas de tempo apresentadas em estações de montagem de componentes em indústrias automotivas.

Figura 23 – Matriz de janelas de tempo

```
data["time_windows"] = [
    (0, 50),   # depot
    (10, 30),  # Estação 1
    (75, 85),  # Estação 2
    (15, 60),  # Estação 3
    (0, 40),   # Estação 4
    (30, 250), # Estação 5
    (80, 95),  # Estação 6
    (5, 120),  # Estação 7
    (5, 120),  # Estação 8
    (5, 120),  # Estação 9
    (0, 60),   # Estação 10
    (0, 50),   # Estação 11
    (0, 40),   # Estação 12
    (5, 60),   # Estação 13
    (0, 80),   # Estação 14
    (0, 70),   # Estação 15
    (1, 80)    # Estação 16
]
```

Fonte: Elaboração própria (2023).

Após a execução do algoritmo, serão obtidos os seguintes *outputs*: a distância total percorrida na rota ótima (aquela que representa a minimização da distância), os nós percorridos na rota ótima (sequência de nós correspondentes aos locais de visita dos *tow trains* durante a entrega de todos os *trolleys* solicitados), o número de *tow*

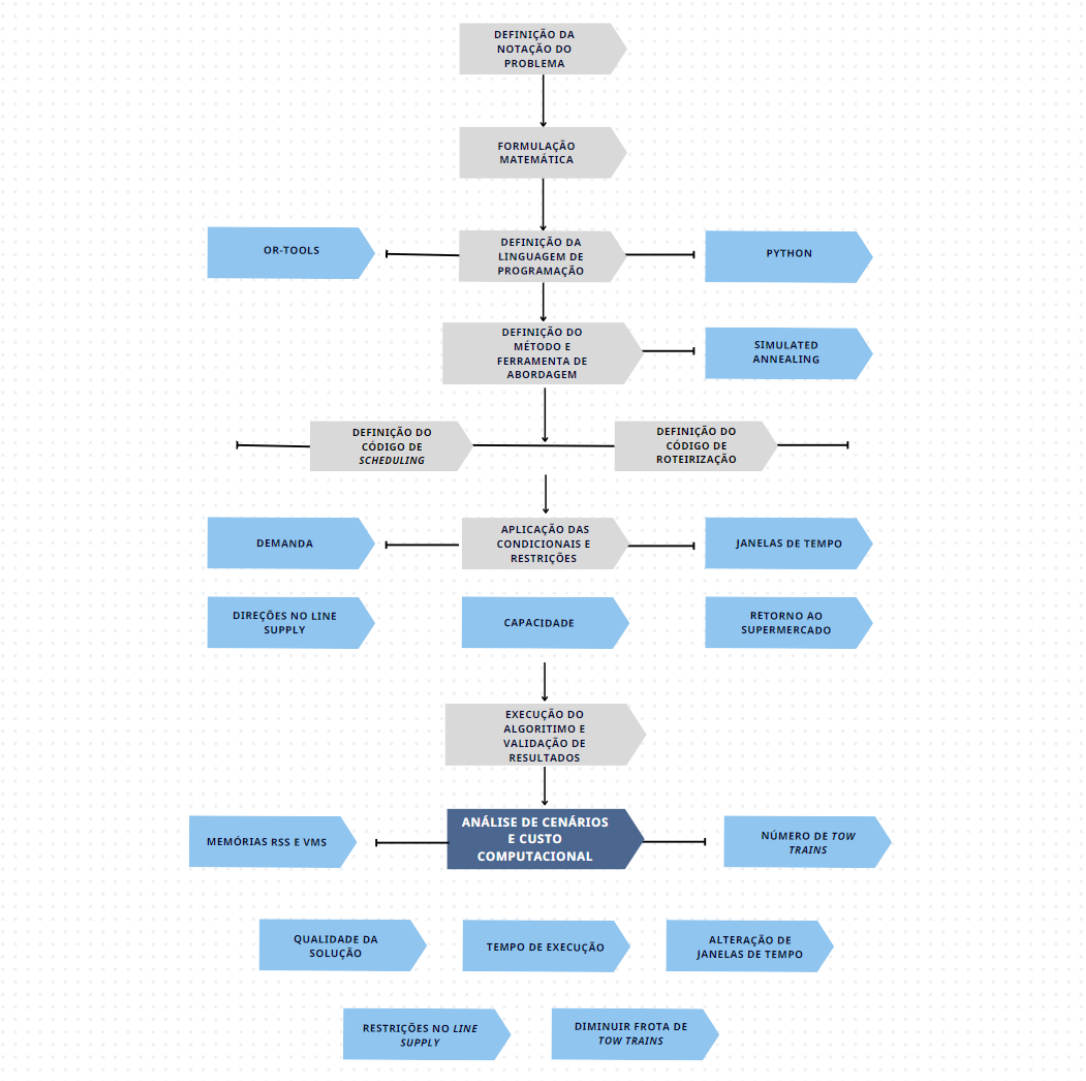
trains utilizados para realizar o transporte esperado, o qual deve ser minimizado, o tempo de execução do algoritmo e o uso das memórias RSS e VNS.

Destaca-se, que os *inputs* mencionados nesta seção correspondem ao cenário inicial a ser simulado. Os demais cenários apresentarão alterações nos parâmetros de inserção, as quais serão descritas em etapas posteriores do presente trabalho.

3.5 DESENVOLVIMENTO DO ALGORITMO

Esta seção apresenta as etapas desenvolvidas ao longo da implementação do algoritmo de roteirização, visando explicitar os processos de pré-concepção, concepção e validação do código proposto. A Figura 24 exibe o fluxograma da execução das etapas contidas no processo de desenvolvimento do algoritmo.

Figura 24 – Fluxograma das etapas de desenvolvimento do algoritmo



Fonte: Elaboração própria (2023).

3.5.1 Notação do problema

O problema de roteirização aqui considerado inicia-se com a notação matemática que regerá a sequência de nós a serem percorridos pelos *tow trains*, durante a execução da rota de abastecimento de peças nas linhas de montagens. Desse modo, assume-se que N é o conjunto de nós e $x_{i,j}$ a variável que indica a conexão entre dois nós i e j inclusos na rota, conforme a Equação 4:

$$x_{i,j} \in \{0,1\} \ \ \forall \ \ i \in N, \ \forall \ \ j \in N \tag{4}$$

Nessa, aponta-se que os parâmetros i e j representam os nós que serão diretamente conectados durante a realização da rota. Ainda, agrega-se à variável $x_{i,j}$ a função de uma variável de decisão binária, de modo que essa iguala-se a 1 caso o *tow train* trace uma rota direta entre as estações indicadas por i e j, ou a 0 caso essa condição não se realize.

Tabela 5 – Índices e parâmetros

Parâmetro	**Descrição**
N	Número total de estações
$d_{i,j}$	A distância entre a estação i e a estação j.
$x_{i,j}$	A variável de decisão binária $x_{i,j}$ = 1, se o *tow train* viaja diretamente da estação i para a j $x_{i,j}$ = 0, caso contrátio
y_k	A variável de decisão binária y_k = 1, se o *tow train* é utilizado na rota y_k = 0, caso contrátio
q_j	A demanda da estação j
a_j	O primeiro horário que o *tow train* pode chegar à estação j
K	O número máximo de *tow trains* disponíveis
Q_i	Capacidade máxima dos *trolleys*
u_i	Instante em que o tow train chega na estação
b_j	Último instante em que o *tow train* pode chegar na estação j
d_{ij}	Tempo de viagem entre estações
C	A capacidade máxima dos *tow trains*

Fonte: Elaboração própria (2023).

Elabora-se a Tabela 5, a qual contém os índices e conjuntos a serem utilizados na etapa de equacionamento, bem como sua descrição.

3.5.2 Formulação matemática

A etapa de formulação matemática inicia-se com a definição do objetivo principal a partir da resolução do problema, sendo esse a minimização da distância total percorrida pelos *tow trains* e do número de veículos usados. A fim de combinar ambos objetivos em um único problema de otimização, utiliza-se uma abordagem multiobjetivo baseada no parâmetro " α", como observado na Equação 5.

$$MIN \quad \alpha \cdot \sum_{i=1}^{N} \sum_{j=1, j \neq i}^{N} d_{i,j} \cdot x_{i,j} + (1 - \alpha) \cdot \sum_{k=1}^{K} y_k \tag{5}$$

Nessa, α será uma variável com valor intermediário entre 0 e 1, a qual é ajustada para controlar a importância relativa de minimizar a distância ou minimizar o número de veículos usados. Desse modo, ao admitir-se valores maiores para a variável, prioriza-se a minimização da distância, enquanto valores menores priorizam a minimização do veículo.

A primeira parte da Equação 5 (parcela referente à primeira contribuição de α) é responsável por representar o objetivo de minimizar a distância total percorrida pelos *tow trains*, somando as distâncias entre todos os pares de nós (i e j), multiplicadas pelas variáveis binárias de decisão x_{ij}, referentes aos pares de nós incluídos na rota.

Já a segunda parte da Equação 5 (parcela referente à segunda contribuição de α) objetiva minimizar o número de *tow trains*, utilizados para realizar o transporte das peças programadas em cada turno de funcionamento da montadora. Desse modo, a somatória é utilizada para iterar entre todos os veículos disponíveis na frota de *tow trains*. A função de mínimo busca encontrar o menor valor entre as somas das variáveis binárias y_k, sendo y_k uma variável de decisão que indica se o *tow train* k foi utilizado ou não. Se $y_k = 1$, significa que o *tow train* k foi de fato utilizado; se $y_k = 0$, ele não foi usado. Dessa forma, a expressão MIN $\sum_{k=1}^{K} y_k$ otimiza a quantidade mínima de *tow trains* a serem empregados para atender a demanda, considerando as restrições do problema.

Algumas restrições devem ser consideradas na construção do modelo:

- Cada nó (estação) deve ser visitado apenas uma vez durante a execução de uma rota:

$$\sum_{j=1, j \neq i}^{N} x_{i,j} = 1 \quad for \;\; i = 1, 2, ..., N \tag{6}$$

- Cada nó (estação) deve ser deixado apenas uma vez durante a execução de uma

rota:

$$\sum_{i=1,i\neq j}^{N} x_{i,j} = 1 \quad for \;\; j = 1, 2, ..., N \tag{7}$$

- Restrição de capacidade/demanda das estações:

$$\sum_{j=1,j\neq i}^{N} q_j \cdot x_{ij} \leq Q_i \quad for \;\; i = 1, 2, ..., N \tag{8}$$

- Restrições de janela de tempo:

$$a_j + d_{ij} \cdot x_{ij} \leq u_i \leq b_j - (d_{ji} - d_{ij}) \cdot x_{ij} \;\; for \;\; i, j = 1, 2, ..., N \tag{9}$$

- Restrições de atribuição de *tow trains*:

$$\sum_{i=1}^{N} x_{ij} = y_k \;\; for \;\; j = 1, 2, ..., N \tag{10}$$

- Restrições de capacidade de *tow trains*:

$$\sum_{i=1,i\neq j}^{N} q_j \cdot x_{ij} \leq C \cdot y_k \quad for \;\; j = 1, 2, ..., N \tag{11}$$

Com isso, destaca-se que as Equações 6 e 7 são responsáveis por garantir que o *tow train* visite e saia de cada estação exatamente uma vez, formando uma rota adequada desde o momento do abastecimento no super mercado, visita à linha de produção e retorno ao supermercado. Nessas, as condições $j \neq i$ e $i \neq j$ são adicionadas para garantir que uma mesma estação (nó) não seja inclusa duas vezes no somatório.

Já a Equação 8 contém o termo $\sum_{j=1,j\neq i}^{N} q_j \cdot x_{ij}$ que representa a soma ponderada das demandas das estações j conectadas à estação i por meio das variáveis de decisão binárias x_{ij}. Cada termo $q_j \cdot x_{ij}$ denota a demanda da estação j se ela estiver incluída na rota do *tow train* que visita a estação i. A variável x_{ij} assume valores de 0 ou 1, indicando se a estação j é conectada à estação i na rota do veículo (x_{ij} = 1) ou não (x_{ij} = 0).

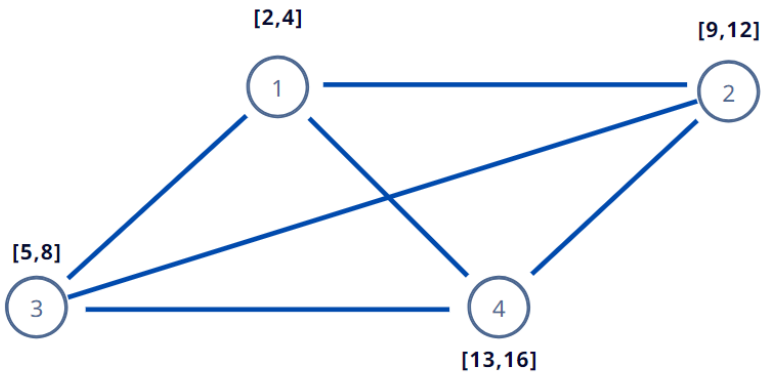
Ainda, indica-se que Q_i representa a capacidade máxima do *tow train* ao visitar a linha de produção, ou seja, o número máximo de *trolleys* que o veículo pode transportar em uma rota. A inequação, então, afirma que a soma das demandas das estações conectadas à estação i não pode exceder a capacidade máxima do *tow train* que a visita, ou seja, a capacidade do veículo deve ser suficiente para atender à demanda das estações em sua rota. A condição $j \neq i$ é adicionada para garantir que uma estação não seja contada duas vezes, evitando duplicatas no somatório.

A Equação 9 é utilizada para garantir que os *tow trains* cheguem a cada estação dentro da janela de tempo especificada, levando em consideração os tempos de viagem.

Além dos parâmetros explicados anteriormente, a formulação conta com os termos u_i (representa o instante em que o veículo realmente chega à estação i); a_j (o primeiro horário permitido para um veículo chegar à estação j); b_j (o último horário permitido para um veículo chegar à estação j); d_{ij} (tempo de viagem entre as estações i e j); e d_{ij} (tempo da viagem de retorno entre as estações j e i) .

Assim, o lado esquerdo da equação ($a_j + d_{ij} \cdot x_{ij}$) calcula o tempo de chegada mais cedo possível na estação i com base no início mais cedo possível na estação j e do tempo de viagem. Já o lado direito da equação ($b_j - (d_{ji} - d_{ij}) \cdot x_{ij}$) garante que o *tow train* saia da estação i antes do final da janela de tempo da estação j, somando os tempos de viagem. O esquema apresentado na Figura 25 permite a visualização desse equacionamento.

Figura 25 – Visualização das janelas de tempo



Fonte: Elaboração própria (2023).

Nessa Figura 25, as janelas de tempo de cada nó são indicadas entre parênteses e, desse modo, os termos a_j e b_j correspondem ao primeiro e segundo termos do parênteses, respectivamente ([2 e 4], por exemplo). Com isso, denota-se que u_i deve ser maior que a_j e menor que b_j durante a realização da rota, considerando o tempo de viagem entre estações. Uma possibilidade de rota que atende as janelas de tempo pode ser definida como 1-3-2-4 com u_i igual a [3, 5, 10, 14].

A Equação 10 é aplicada a cada *tow train* k, garantindo que a soma das rotas x_{ij} para cada estação i seja igual a y_k, variável binária que indica se o veículo k é usado. Assim, define-se que, se o *tow train* k for usado ($y_k = 1$), as rotas atribuídas a esse veículo (a soma de x_{ij} para todas as estações i correspondem à sua disponibilidade. Se $y_k = 0$, os *tow trains* não estão sendo utilizados e suas rotas atribuídas serão todas zero.

Por fim, a Equação 11 possui definição semelhante à Equação 8, diferenciando-

se, apenas, por agregar termos que referem-se à capacidade dos *tow trains*.

3.5.3 Linguagem Python

Algumas das vantagens dessa linguagem, que justificam a adoção para confecção do código utilizado no presente trabalho (Python 3), podem ser apontadas como um vasto conjunto de bibliotecas e ferramentas, vasta documentação e sintaxe simples, clara e expressiva. Ademais, esta possui orientação a objetos (POO), o que torna sua utilização mais simplificada (CABRAL, 2013).

Na POO, foca-se na resolução de problemas de instâncias inferiores para, assim, chegar em uma solução completa. Adota-se, deste modo, no código os seguintes conceitos de programação orientada a objetos: Classes, as quais definem-se como descrição de um modelo que especifica as propriedades e o comportamento para objetos similares e Objeto, definido como a instância gerada a partir de uma mesma classe (MACHADO; FRANCO; BERTAGNOLLI, 2016).

Em virtude da evolução da tecnologia desde a sua criação, a linguagem Python não contém todas as ferramentas necessárias para a programação atual. Assim, inserem-se as bibliotecas as quais contêm rotinas computacionais que, para sua utilização, precisam ser instaladas em paralelo com a importação da rotina desejada (MALDANER, 2019).

Para o presente trabalho de conclusão de curso, optou-se por utilizar a biblioteca NumPy que, de acordo com Oliphant (2007), possui como principal função o fornecimento de um dado nomeado arranjo, isso é, utiliza um novo padrão de organização de dados para a resolução de questões envolvendo matrizes (arranjos bidimensionais) ou vetores (arranjos unidimensionais), o qual facilita o trato dos problemas de otimização.

Ainda, adicionou-se a biblioteca psutil a fim de medir os tempos de execução e a utilização de memória do algoritmo. Esta biblioteca fornece uma interface simples para acessar informações sobre a utilização de recursos do sistema, como CPU, memória e disco, possibilitando uma análise detalhada do desempenho do algoritmo em diferentes cenários.

Por fim, para facilitar a análise e interpretação dos resultados obtidos, a biblioteca pandas foi incorporada ao projeto. A escolha dessa biblioteca se baseia em sua capacidade robusta de manipular e analisar dados, visto que essa oferece estruturas de dados, como o DataFrame, que simplificam a organização e a exportação de resultados para diversos formatos, incluindo o Microsoft Excel.

3.5.4 OR-Tools

O OR-Tools, desenvolvido pela companhia Google, é um software de código aberto voltado para a otimização combinada, caracterizado pela busca da melhor solução para um problema em um conjunto grande de soluções possíveis. Entre tais problemáticas pode-se citar o roteamento de veículos (foco principal do presente trabalho), fluxos, programação linear ou programação inteira mista (ALDEANO, 2020).

A fim de obter as soluções ótimas globais ou locais nas questões de otimização, este software utiliza algoritmos de última geração para restringir a exploração do espaço de solução, incluindo diferentes heurísticas, meta-heurísticas e métodos de busca. Para tal, ainda que escrito inicialmente na linguagem C++, o OR-Tools possui suporte para as linguagens de programação Python, Java ou .Net (PERRON; FURNON, 2023).

Em virtude do aumento do custo computacional, frente à adoção de parâmetros de entradas complexos ou extensos (capacidades de transporte, sistemas penalidades, tempos específicos de retiradas, entre outros), torna-se inviável a utilização única dessa biblioteca para a solução do problema aqui proposto (VERCELLINO, 2020). Sugere-se, portanto, a integração das ferramentas ofertadas por tal software com as demais bibliotecas mencionadas nesta Seção metodológica (biblioteca padrão Python e NumPy).

3.5.5 Simulated Annealing

Em virtude das vantagens mencionadas na Seção 2.5 acerca da utilização de meta-heurística para a resolução de problemas de roteirização aplicados em linhas de montagem de montadoras de automóveis, propõe-se a adoção, nesse trabalho, da meta-heurística *Simulated Annealing*

Para tal, segue-se com o procedimento inicial de aplicação do *Simulated Annealing*, no qual uma solução genérica inicial é adotada (durante o desenvolvimento do código a solução inicial é definida como a primeira a ser obtida após execução das restrições e condicionais aplicadas, ou seja, uma dada solução qualquer). Com isso, aplica-se, em sequência, funções de perturbação e avaliação para que o algoritmo execute novas buscas de soluções até alcançar o critério de parada pré-definido (GOMES, 2003).

A etapa de definição da solução inicial conta com uma restrição baseada no processo de busca de novas soluções: se a heurística de construção (parcela do código responsável por admitir o cálculo da solução base) não conseguir encontrar uma solução viável, o algoritmo SA termina, definindo o primeiro critério de parada.

Destaca-se, que a partir da execução, a cada ciclo, o algoritmo irá comparar o custo final da solução com o melhor atual, de modo que, caso o primeiro for melhor que o último, salva-se o melhor atual e prossegue-se para o próximo ciclo até que as

condições de parada sejam atendidas. Infere-se que os parâmetros para considerar uma solução mais adequada do que outra dão-se pela adição no código dos objetivos operacionais mencionados nas seções anteriores do presente trabalho: reduzir a distância total percorrida e reduzir a frota de *tow trains*.

Garantida a obtenção de uma solução inicial, permite-se que essa seja perturbada ao longo da execução do *loop* principal do algoritmo, de modo que a cada iteração das estações contidas na linha (i = 0 até $i = N$) o algoritmo escolhe um novo nó aleatório para a reexecução do *loop* e obtenção da solução do problema de roteirização. Sendo a nova solução (S_i') melhor do que a solução antiga (S_i), então ela se torna a nova solução atual, caso contrário o algoritmo escolhe um novo nó aleatório, reiniciando o processo de busca.

Em virtude do caráter multiobjetivo do problema de roteirização aqui proposto, mostra-se necessário considerar um peso que determine a importância relativa para a obtenção de cada uma dessas soluções. A redução da frota apresenta-se como objetivo prioritário, devendo possuir, portanto, um peso agregado superior (o valor do peso será quantificado e demonstrado em etapas posteriores).

No que tange aos critérios de parada do algoritmo, admite-se que, além da parada devido a ausência de solução inicial, outras duas restrições são impostas. A primeira relaciona-se com o limite superior para a pausa em cada estação - aqui definido como 240 segundos - (uma vez que os *tow trains* devem parar para descarregar os *trolleys* contendo as peças solicitadas na linha de montagem) e durante a execução de uma rota completa. Essa restrição funciona como uma ferramenta de controle para revelar a presença de erros durante a realização das rotas de entrega de peças, uma vez que, caso alguma condição não permita a realização da entrega, o *tow train* permanece fixo na respectiva estação até que o limite superior de tempo seja atingido, parando a execução do código.

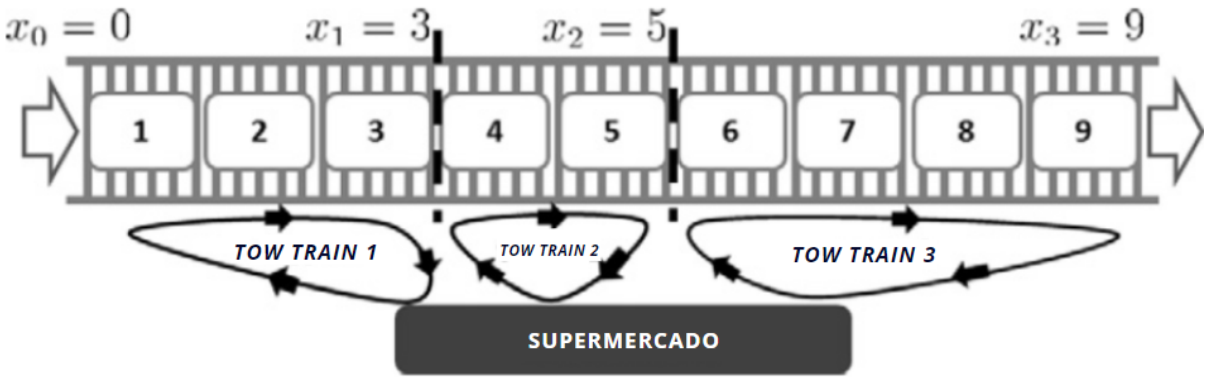
O outro critério de parada acontece quando o algoritmo percorre todos os nós inseridos na matriz de distâncias (representativa das posições do supermercado e das estações na linha de montagem), revelando que todas as condições de roteirização foram realizadas corretamente durante sua execução.

3.5.6 Roteirização

Aponta-se que para a adoção de uma solução flexível e de fácil manipulação, o algoritmo desenvolvido representa o percurso dos *tow trains* por meio de um vetor inteiro que sempre começa e termina no supermercado ($i = 0$), visitando as estações ao longo do percurso prescrito. Com esta representação pode-se desenvolver rotinas paralelas a fim de calcular a distância total de uma rota ou o número de *tow trains* necessários para realizar o reabastecimento de peças nas linhas de produção, ou seja, atender a todos *line supplies* (ALI, 2019).

A abordagem de programação dinâmica adotada para a configuração do cenário de roteirização caracteriza-se por admitir que as rotas realizadas por um dado *tow train* não dependem das rotas realizadas pelos demais, e sim, dos nós presentes à suas esquerdas e direitas (estações próximas e anteriores à sua posição atual) (EMDE; BOYSEN, 2011). Tal afirmação é decorrente da adição de possíveis rotas sobrepostas (dois *tow trains* diferentes podem realizar a mesma rota para entregar *trolleys* diferentes desde que não ocupem um mesmo nó simultaneamente), cenário que pode ser verificado na Figura 26.

Figura 26 – Rotas pré-definidas



Fonte: Adaptado de Emde e Boysen (2011, p. 292).

Com essa representação permite-se adotar uma subdivisão de N nós na rota, onde cada nó de índice $i+1$ denota uma estação consecutiva à outra (em termos de matriz de distâncias), como apresentado na etapa de equacionamento matemático. A fim de percorrer todos os nós inclusos na matriz referente às estações, cria-se um padrão de recursão direta para o algoritmo (processo iterativo utilizado para percorrer todos os nós integrantes da matriz de distâncias a partir da adição de um valor unitário à variável i a cada *loop* finalizado), no qual obtém-se os valores objetivos parciais para transições de um ponto i para outro ponto $i+1$ (o objetivo parcial ocorre em decorrência da aplicação do *loop* principal do algoritmo para todos os nós integrantes da matriz, não exclusivamente dos nós representativos das entregas na linha de montagem, os objetivos primários).

Esse padrão de recursão direta é simplificado por meio da ferramenta OR-Tools, na qual as transições entre estações e o consequente cálculo cumulativo de distâncias percorridas são representadas por um *callback* de distâncias (função que relaciona os nós integrantes da matriz de distâncias e retorna a distância entre eles).

Desse modo, resolve-se o problema do cálculo de distâncias entre os pontos de

uma rota por meio da adição do *callback* e a consequente criação de uma dimensão de distância (levando-se em consideração o ponto inicial de uma rota como sendo o índice do supermercado e iterando entre os demais pontos contidos nas subestações), uma variável que receberá o valor cumulativo das distâncias percorrida por cada veículo ao longo do trajeto.

Com isso, ajustada a ferramenta de cálculo de distâncias entre os pontos a serem percorridos em uma rota, mostra-se necessário realizar os procedimentos para indicar a rota ótima e salvá-la em um vetor, o qual será impresso ao final da solução. Para tal, utiliza-se um avaliador de custo de arco, o qual é responsável por informar ao solucionador como calcular o custo da viagem entre dois pontos, como visto em sequência:

- routing.SetArcCostEvaluatorOfAllVehicles(transitcallbackindex)

Destaca-se que o parâmetro ”transitcallbackindex” atua como avaliador de custo do arco, sendo a referência interna do solucionador para o *callback* de distância.

Adota-se, ademais, para realizar tal solução cumulativa, uma matriz de distâncias, a qual indicará a distância entre cada uma das estações contidas na rota de acordo com o respectivo índice i, j do nó. Tal matriz foi pré-computada a fim de reduzir o custo computacional de execução do algoritmo, de modo que necessita-se apenas dos índices (i e j) para identificação das estações abordadas.

Concluídas as etapas de implementação desses solucionadores, é possível realizar a implementação da solução de pesquisa e do método heurístico para encontrar as primeiras soluções. Desse modo, a solução ótima de roteirização é representada pela rota que percorre todos os nós indicados pelo operador por meio da minimização do parâmetro ”solution”, variável na qual os dados de custos totais entre nós serão armazenados. Esse procedimento pode ser exemplificado pela Figura 27.

Figura 27 – Implementação da solução de pesquisa e heurística

```
# Setting first solution heuristic.
search_parameters = pywrapcp.DefaultRoutingSearchParameters()
search_parameters.first_solution_strategy = (
    routing_enums_pb2.FirstSolutionStrategy.PATH_CHEAPEST_ARC
)

# Solve the problem.
solution = routing.SolveWithParameters(search_parameters)

# Print solution on console.
if solution:
    print_solution(manager, routing, solution)
```

Fonte: Elaboração própria (2023)

Desse modo, o algoritmo base para a realização da roteirização é finalizado, possibilitando a inserção, em sequência, dos parâmetros que envolvem a realização do *scheduling* e das restrições ao processo de reabastecimento das linhas de produção.

3.5.7 Scheduling

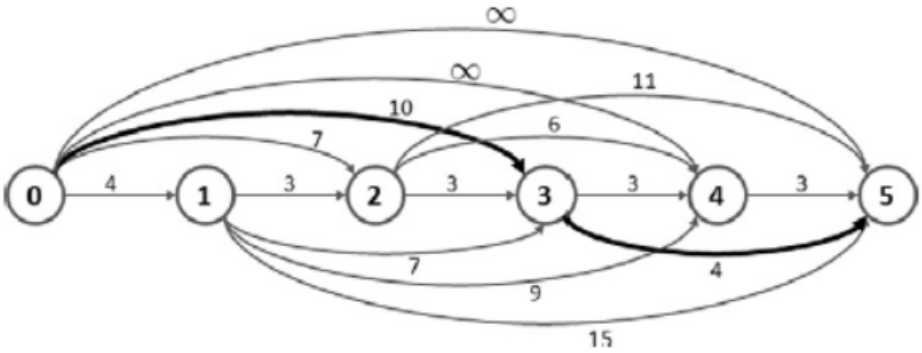
Nessa seção serão definidos os requisitos necessários para realização do *scheduling* dos *tow trains*, ou seja, deve-se garantir os requisitos da programação de operação de cada *tow train*, como o número de rotas necessárias para garantir toda a operação de reabastecimento da linha de montagem, levando-se em consideração a programação desde o momento de carregamento no supermercado, até a realização do ciclo completo por todas as estações atribuídas. A etapa de *schedule* também leva em consideração as janelas de tempo específicas de cada estação, de modo que deve-se garantir que os *tow trains* visitem as estações nos intervalos de tempo adequados.

Adota-se, em fins de simplificação, que os *tow trains* levam um tempo constante (Equação 12) para percorrer todas as estações em sua rota e ser reabastecido no supermercado, sendo que tal janela de tempo baseia-se no tempo necessário para o *tow train* chegar ao supermercado após a última estação j, ser reabastecido e chegar à primeira estação i da próxima rota (P_{ij}) (tempo equivalente à realização da rota completa) e no tempo necessário para o *tow train* percorrer a distância entre uma estação i e j (p_{ij}) (EMDE; BOYSEN, 2011).

$$D = p_{ij} + P(i,j) \tag{12}$$

Adiciona-se, em paralelo, a restrição de que os que os *tow trains* não são capazes de atender uma estação por mais de uma vez antes de seu retorno ao supermercado e, portanto, as rotas são independentes umas das outras, visando simular o ambiente atual de uma montadora no qual os ciclos de entrega (*waves*) são realizados em uma única rota. Desse modo, o problema de *scheduling* irá consistir em dividir uma matriz de C ciclos (viagens entre estações) em intervalos distintos, cada um com pelo menos um comprimento D (tempo igualitário de viagem entre estações).

Figura 28 – Realização de ciclos



Fonte: Adaptado de Emde e Boysen (2011, p. 292).

Representa-se o problema, desse modo, pela inserção de uma matriz de janelas de tempos; uma matriz de tempos médios de viagens (previamente computada) e o número máximo de veículos presentes na frota de *tow trains*. Em sequência, possibilita-se inserir uma função que cria um *callback* de tempo (Figura 29) e o transmite para o solucionador, em paralelo com a definição dos custos do arco (parâmetro que define o custo da viagem, como sendo os tempos de viagem entre os locais).

Figura 29 – *callback* de tempo

```
def time_callback(from_index, to_index):
    """Returns the travel time between the two nodes."""
    # Convert from routing variable Index to time matrix NodeIndex.
    from_node = manager.IndexToNode(from_index)
    to_node = manager.IndexToNode(to_index)
    return data["time_matrix"][from_node][to_node]

transit_callback_index = routing.RegisterTransitCallback(time_callback)
routing.SetArcCostEvaluatorOfAllVehicles(transit_callback_index)
```

Fonte: Elaboração própria (2023).

De maneira semelhante ao acúmulo de distâncias percorridas pelos *tow trains* durante a implementação do código de roteirização, inclui-se um código a fim de criar uma dimensão para o tempo de viagem dos veículos. Essas dimensões são responsáveis por rastrear as distâncias acumuladas ao longo da rota de um veículo. Ainda, adiciona-se um limite superior para a pausa (tempos de espera nas estações) e um limite superior para o tempo total em cada trajeto do veículo, visando eliminar erros decorrentes de pausas inesperadas ou muito longas, como exemplo pode-se adotar que o tempo máximo para execução de uma rota completa seja equivalente a 30 minutos e, caso esse seja ultrapassado, define-se a parada da simulação.

Por fim, cria-se a restrição de visita nos nós apenas nos momentos permitidos

pelas janelas de tempo indicadas pelos usuários, permitindo que os resultados sejam computados e analisados por meio das mesmas estratégias de buscas definidas anteriormente. Com isso, o algoritmo poderá retornar as janelas de solução (Figura 30), as quais indicarão o intervalo de tempo em que um *tow train* precisa chegar a um determinado nó para atender à programação.

Figura 30 – Implementação do retorno das janelas de tempo

```
            index = solution.Value(routing.NextVar(index))
        time_var = time_dimension.CumulVar(index)
        plan_output += (
            f"{manager.IndexToNode(index)}"
            f" Time({solution.Min(time_var)},{solution.Max(time_var)})\n"
        )
        plan_output += f"Time of the route: {solution.Min(time_var)}min\n"
        print(plan_output)
        total_time += solution.Min(time_var)
    print(f"Total time of all routes: {total_time}min")
```

Fonte: Elaboração própria (2023).

3.5.8 Restrições de demanda e capacidade

A fim de adicionar as restrições de demanda de cada estação de trabalho disposta ao longo da linha de montagem, às capacidades máximas dos *trolleys* a serem transportados pelos *tow trains* e da quantidade máxima de *trolleys* que podem ser atrelados aos *tow trains*, mostra-se necessário incluir parâmetros restritivos fixos no algoritmo. Para tal, inclui-se três vetores ao código, sendo esses referentes às demandas das estações, a capacidade dos *trolleys* e a capacidade dos *tow trains*, como exemplificado na Tabela 6.

Tabela 6 – Exemplo de vetores

Vetores	**Descrição**
data["demandasestacoes"] = [0, 1, 1, 2, 4, 8, 8]	Vetor de demanda das estações
data["towtrainscapacidades"] = [5, 5, 5, 5]	Vetor de capacidade dos *tow trains*
data["trolleyscapacidades"] = [6, 6, 6, 6]	Vetor de capacidade dos *trolleys*

Fonte: Elaboração própria (2023).

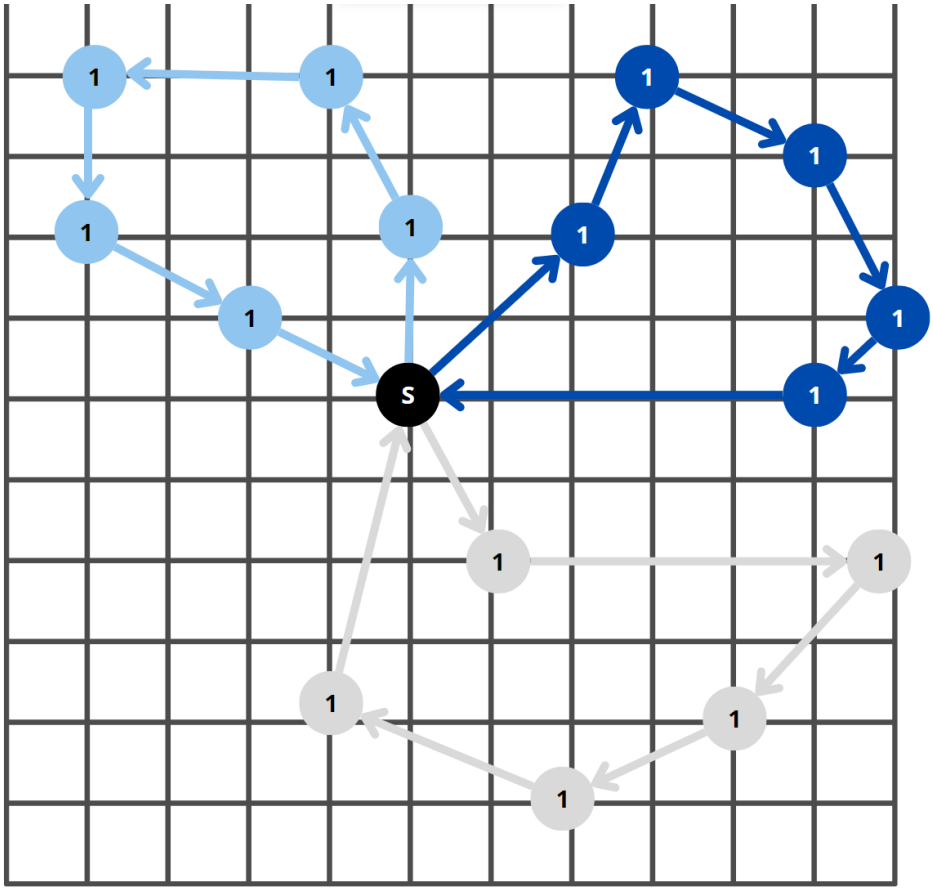
Desse modo, as seguintes restrições são impostas ao algoritmo:

- Demandas: cada local tem uma demanda correspondente à quantidade (por exemplo, peso ou volume) do item a ser entregue.
- Capacidades: cada *tow train* e *trolley* possuem uma capacidade (os *tow trains*

possuem uma quantidade máxima de *trolleys* que podem ser inseridos no comboio e os *trolleys* possuem um limite de peças a serem transportadas). À medida que um veículo viaja ao longo da rota, a quantidade total de itens que ele pode levar nunca poderá exceder sua capacidade.

Prossegue-se com a adição de um *callback* de demanda para o solucionador, a qual retornará a demanda existente em cada nó e uma dimensão refente às restrições de capacidade. Um exemplo da aplicação do *callback* de demanda pode ser visto na Figura 31.

Figura 31 – Exemplo prático do *callback* de demanda



Fonte: Elaboração própria (2023).

Nesse exemplo, considerando-se um *tow train* com capacidade máxima igual a 5 *trolleys*, mostram-se 3 possibilidade (representadas pelas setas azul claro, azul escuro e cinza) de atendimento das demandas unitárias. Assim, demonstra-se que apenas 5 estações podem ser atendidas antes do *tow train* retornar ao supermercado (nó central representado pela cor preta) para reabastecer.

Desse modo, após implementação dos vetores de demanda e capacidade, bem como, dos *callbacks* de demanda e distância, o algoritmo adquire a capacidade

de atender condicionais às restrições de demanda e capacidade pré-definidos. Esses *callbacks* permitem que o algoritmo avalie a capacidade máxima dos *trolleys* e *tow trains* e a demanda de cada estação visitada, assegurando que a rota gerada respeite as restrições de carga e atenda às exigências de cada estação.

3.5.9 Restrições de janela de tempo

As restrições de janela de tempo são inseridas no algoritmo a fim de englobar os casos em que as visitas dos tow trains para as estações de trabalho ao longo da linha de montagem envolvem a programação disponível apenas durante períodos específicos. Levando-se em consideração que a programação fabril tende a ser realizada com antecedência, as viagens diárias ocorrem de acordo com um cronograma restrito e pouco sujeito a alterações, de modo que, espera-se que as janelas de tempo sejam atendidas fielmente ao longo da programação (EMDE; BOYSEN, 2011).

O desenvolvimento do algoritmo deve contar, então, com a adição de uma matriz de tempos de janelas de tempos, a qual contém os perídos de agendamento de visitas. Desse modo, os dados constituintes da adição devem englobar as seguintes matrizes e vetores:

Desse modo, os seguintes condicionais são impostos ao algoritmo (Tabela 7):

Tabela 7 – Dados para janelas de tempos

Item	**Descrição**
data[$janela_{tempo}$]	Matriz de janelas de tempo que podem ser consideradas horários para uma visita.
data[$matriz_{tempo}$]	Matriz de tempos de viagem entre locais.
data[$num_{towtrain}$]	Número de *tow trains* na frota
data[depot]	O índice do supermercado.

Fonte: Elaboração própria (2023).

Prossegue-se com a adição de um *callback* referente aos horários de visitas por meio de uma função responsável pela criação do *callback* de tempo e sua transmissão para o solucionador (Figura 32). A função também define os custos do arco, que definem o custo da viagem, como sendo os tempos de viagem entre os locais.

Figura 32 – Restrição de janela de tempo

```
time = "Time"
routing.AddDimension(
    transit_callback_index,
    30,  # tempo de espera
    30,  # tempo máximo por veículo
    False,  # Não força o início para 0
    time,
)
time_dimension = routing.GetDimensionOrDie(time)

# Adiciona restrição de janela de tempo para as estações

for location_idx, time_window in enumerate(data["time_windows"]):
    if location_idx == data["depot"]:
        continue
    index = manager.NodeToIndex(location_idx)
    time_dimension.CumulVar(index).SetRange(time_window[0], time_window[1])

# Adiciona restrição de janela de tempo para os veículos no nó de entrada

depot_idx = data["depot"]
for vehicle_id in range(data["num_vehicles"]):
    index = routing.Start(vehicle_id)
    time_dimension.CumulVar(index).SetRange(
        data["time_windows"][depot_idx][0], data["time_windows"][depot_idx][1]
    )
for i in range(data["num_vehicles"]):
    routing.AddVariableMinimizedByFinalizer(
        time_dimension.CumulVar(routing.Start(i))
    )
    routing.AddVariableMinimizedByFinalizer(time_dimension.CumulVar(routing.End(i)))
```

Fonte: Elaboração própria (2023).

Por fim, adicionadas as restrições de janela de tempo, o algoritmo torna-se capaz de garantir as condições de entrega pré-estabelecidas. Após a conclusão da inserção de todas as restrições mapeadas, possibilitou-se finalizar a etapa de implementação do código para resolução do problema de roteirização e *scheduling*.

3.5.10 Estratégias para criação de cenários

Pode-se apontar que a análise comparativa de cenários é uma abordagem fundamental para avaliar a eficiência de soluções propostas em modelos de otimização. Isso pois, ao incorporar diferentes cenários permite-se uma avaliação completa e abrangente das respostas retornadas pelo modelo heurístico proposto à medida que esse é submetido a situações variáveis de condições operacionais.

Em virtude de tal relevância, propõe-se, para o presente trabalho, comparar os resultados referentes às seguintes estratégias de criação de cenários para o código desenvolvido:

- Diminuir frota máxima de *tow trains*
- Tornar todas as ruas com idas e vindas (mão dupla)

- Aumentar as janelas de tempo
- Reduzir as janelas de tempo

A aplicação da estratégia 1 (Diminuir frota máxima de *tow trains*) justifica-se à medida que sua utilização permite explorar a eficiência da solução em termos de custos operacionais e utilização de recursos. Ademais, ao diminuir a frota máxima realiza-se uma diminuição dos custos agregados à sua manutenção, porém há a possibilidade de reduzir-se o atendimento a linha de montagem, fator prejudicial.

No que tange à estratégia de tornar todas as ruas da linha de montagem de mão dupla, espera-se criar um impacto na conectividade das rotas e a eficiência do trajeto. Já aumentar e reduzir as janelas de tempo permite avaliar a sensibilidade do modelo a restrições temporais e, em paralelo, testar até em que ponto as operações dos *tow trains* poderão ser manipuladas em termos temporais.

Destaca-se, ainda, que a fim de simular os cenários, necessita-se realizar pequenas modificações no código origem para englobar os diferentes parâmetros de *input*. Ainda assim, considerando-se que as modificações concentram-se em variáveis *inputadas* pelo usuário ou fixas, variáveis de fácil alteração e manipulação, não esperam-se grandes alterações na estrutura do código e, portanto, tais modificações não serão profundamente explicitadas.

Através dessas comparações, por fim, é possível identificar as limitações e os pontos fortes da solução proposta em diferentes contextos, auxiliando em processos de tomada de decisões e na seleção da melhor abordagem para situações específicas apresentadas na montadora.

3.5.11 Custo computacional

No presente estudo, a avaliação do custo computacional do algoritmo desenvolvido para otimizar o processo de roteirização e *scheduling* de *tow trains* em linhas de montagem de automóveis é abordada a partir da comparação entre duas meta-heurísticas: vizinho mais próximo e *simulated annealing*. Essa avaliação é fundamental para compreender a eficácia e a viabilidade prática da aplicação do algoritmo (desenvolvido em primeiro plano com um solucionador baseado no método SA) em um ambiente de manufatura automobilística altamente competitivo.

Para uma análise abrangente e detalhada do algoritmo, foram definidas métricas de avaliação, incluindo o tempo de execução (TE), a qualidade da solução de roteirização (QS), o *Resident Set Size* (RSS), o *Virtual Memory Size* (VMS) e o número de *tow trains* utilizados (TTU). Essas métricas podem ser descritas como:

- Tempo de Execução (TE): Essa métrica é de importância crucial, uma vez que reflete a rapidez com que cada algoritmo gera soluções. No contexto de montadoras automobilísticas, a tomada de decisões precisa ocorrer em tempo

real para otimizar a produção, justificando a validade do parâmetro. No presente algoritmo, o tempo de execução é medido em segundos.

- Qualidade da Solução (QS): Para avaliar a qualidade das soluções geradas, utiliza-se a distância total percorrida pelos *tow trains*, sendo que quanto menor for a distância, mais eficiente é a solução. Essa métrica é expressa como uma porcentagem da diferença entre as distâncias das soluções geradas pelo solucionador com as diferentes meta-heurísticas.
- *Resident Set Size* (RSS): O RSS representa a quantidade de memória física (RAM) que um processo está atualmente usando, assim, este parâmetro destaca-se como uma medida da memória real que um processo está ocupando no momento e é uma métrica importante para determinar o uso atual de memória no algoritmo com as diferentes meta-heurísticas.
- *Virtual Memory Size* (VMS): O VMS representa a quantidade total de memória virtual que um processo tem alocada, incluindo tanto a memória na RAM quanto na memória de troca (swap) e outros recursos de armazenamento.
- Número de *tow trains* Utilizados (TTU): O número de *tow trains* utilizados é um indicador crítico, pois está diretamente relacionado à economia de recursos gerada pelo algoritmo. Essa métrica é expressa como uma porcentagem da diferença entre os números de *tow trains* utilizados nas soluções geradas pelo solucionador com as diferentes meta-heurísticas.

A comparação entre os algoritmos, guiada por esses cinco critérios, fornecerá uma visão abrangente da eficiência e aplicabilidade dos solucionadores (representados pela parcela do algoritmo desenvolvido com a aplicação da heurística Vizinho mais próximo e a heurística *Simulated Annealing* como métodos de obtenção das soluções objetivadas) no processo de otimização da roteirização e *scheduling* de *tow trains* em linhas de montagem de automóveis. Essa análise, então, permite a identificação da meta-heurística mais viável para integrar o algoritmo e adequar-se aos ambientes de produção das montadoras.

De acordo com o mencionado em seções anteriores, os testes aplicados ao algoritmo foram conduzidos em cenários variados, representando diferentes níveis de complexidade do reabastecimento das linhas de montagem, a fim de simular um funcionamento realista desses ambientes produtivos. A metodologia de comparação dos algoritmos deu-se com a criação de uma tabela de comparação que apresenta as métricas de avaliação para cada um dos cenários avaliados com as diferentes meta-heurísticas aplicadas no solucionador.

A importância dessa abordagem tabelada reside na possibilidade de fornecer uma visão sistêmica e estruturada dos resultados obtidos, uma vez que esses são extensos em virtude do alto número de cenários simulados. A análise comparativa tabelada permite, assim, a avaliação do desempenho do algoritmo sob diferentes

critérios de avaliação, priorizando-se as métricas anteriormente mencionadas.

Ademais, a comparação contribui para a identificação de tendências e padrões nos resultados, possibilitando a análise das relações entre as métricas TE, QS, TTU, RSS e VMS e as meta-heurísticas aplicadas. Essa análise cruzada é essencial para compreender como as diferentes meta-heurísticas se comportam em cenários específicos, e se há situações em que uma abordagem é mais vantajosa do que a outra.

No capítulo subsequente, serão detalhados os resultados obtidos através das análises de cenários simulados no algoritmo e das comparações de custo computacional com as duas meta-heurísticas pré-definidas. Com a comparação das métricas de análise, tais como Tempo de Execução, Qualidade da Solução, Tempo Total de Utilização e Memórias RSS e VMS, espera-se obter conclusões sobre o desempenho das meta-heurísticas adotadas e do comportamento do algoritmo em cenários específicos. Além disso, espera-se possibilitar a dentificação de tendências e padrões que podem esclarecer quando e por que uma meta-heurística pode superar a outra durante o processo de otimização.

4 RESULTADOS DOS CENÁRIOS SIMULADOS E CUSTO COMPUTACIONAL

Esse capítulo apresenta os resultados obtidos a partir da execução do algoritmo proposto no presente trabalho para realizar o processo de roteirização e *schedule* de *tow trains* em linhas de montagens automotivas. Os resultados dão-se em função da aplicação dos diferentes cenários, restrições e condicionais explicitados nas seções anteriores e serão tabelados a fim de possibilitar a análise cruzada dos dados obtidos.

Também, serão apresentados os resultados referentes à análise comparativa dos custos computacionais apresentados na execução do algoritmo com um solucionador baseado em diferentes meta-heurísticas.

4.1 ANÁLISE DOS CENÁRIOS SIMULADOS E RESULTADOS

Como apresentado na Seção 3.5.10, opta-se por definir 4 estratégias de criação de cenários para delineamento da simulação do ambiente produtivo de uma montadora de automóveis a ser inserido no algoritmo proposto. Esse tópico, então, engloba a caracterização dos cenários e definição dos *inputs* alterados em cada um destes. Em sequência, descreve-se o cruzamento dos resultados obtidos em cada simulação, gerando-se uma tabela que possibilita a análise dos dados priorizados para entendimento do ambiente simulado.

4.1.1 Cenários simulados

Nesta seção, destaca-se que cada um desses cenários a ser simulados foi criado com a intenção de analisar o impacto de diferentes variáveis nas métricas de avaliação do algoritmo. Desse modo, para a promover a adequação do ambiente simulado no algoritmo a cada um dos cenários, dados e *inputs* específicos foram utilizados.

O cenário original, representado como cenário 0, permanece como base fixa de comparação, uma vez que esse representa o cenário típico de montadoras de automóveis, e possui os dados delimitados de acordo com o referido na Seção 3.4.2. O detalhamento dos *inputs* distintos é essencial para compreender o comportamento do algoritmo sob diferentes condições e para a verificação dos possíveis parâmetros mutáveis para otimização de linhas de montagem automotivas, sendo definidos de acordo com as estratégias em sequência:

- Diminuir Frota Máxima de *tow trains*: Nesta estratégia, o foco está na redução do número máximo de *tow trains* utilizados no processo. Os *inputs* são ajustados para avaliar como a diminuição da frota de *tow trains* afeta a distância total percorrida

e a eficiência do agendamento. Levando-se, em conta, que comumente o máximo valor base para a frota é equivalente a 5 veículos, como mencionados em seções anteriores, definiu-se que as variações de cenários contemplariam uma frota com 8, 7, 6, 5, 4 e 3 *tow trains* para cada cenário (foram escolhidos 3 valores superiores e dois inferiores ao adotado no cenário 0 a fim de gerar variabilidade nas análises, identificar o número de *tow trains* limitantes para o atendimento da demanda e verificar como o aumento da frota afeta as distâncias percorridas). Destaca-se que a frota foi reduzida a níveis drásticos (em comparação com o tamanho da linha de montagem aqui simulada) a fim de verificar até em que nível seria possível reduzir o número de veículos sem alterar o processo de reabastecimento das linhas.

- Tornar Todas as Ruas com Idas e Vindas: A característica bidirecional de todas as ruas foi adicionada como alternativa para os cenários com limitações de direções nos *line supplies*. Isso tem o potencial de influenciar o tempo necessário para percorrer cada rota e na distância total percorrida, o que será refletido na comparação com as rotas restritas. Destaca-se, ainda, que a restrição de movimentação nos *line supplies* pode gerar problemáticas para o atendimento das janelas de tempo das estações de trabalho, fato que será evidenciado com a execução do algoritmo.
- Aumentar as Janelas de Tempo: Esta estratégia concentra-se em avaliar os efeitos do aumento das janelas de tempo disponíveis para as estações de trabalho durante o reabastecimento de peças. Isso pode impactar a flexibilidade no agendamento de visitas e na eficiência geral (em termos de tempo decorrido e distâncias percorridas). As novas janelas sugeridas, então, tiveram um aumento de 10 e 50 minutos no intervalo de visitas permitidas, em comparação com o cenário 0.
- Reduzir as Janelas de Tempo: Por fim, a última estratégia explora o oposto da anterior, com a redução das janelas de tempo. Isso pode criar restrições mais rígidas, influenciando a dinâmica do agendamento de visitas e a otimização do número de *tow trains* e distâncias percorridas. Nos cenários, as janelas tiveram uma redução de 5 e 15 minutos no tempo de visita, em comparação com o cenário 0. Destaca-se que essa redução se apresenta como crítica, uma vez que o cenário 0 já possui janelas estreitas (como comumente identificado em montadoras de automóveis).

No que tange aos demais *inputs*, para todos os casos simulados estes mantiveram-se iguais ao aplicado no cenário 0, como destacado na Seção 3.5.10.

A aplicação conjunta dos cenários e *inputs* definidos permite, em consequência, a estruturação de duas tabelas comparativas (Tabelas 8 e 9), que auxiliam na análise dos dados obtidos com a computação do algoritmo e na consequente análise do desempenho apresentado pela ferramenta,

Tabela 8 – Tabela de comparação

Janelas de tempo	Tow trains	Ruas	Meta-heurística	Janelas de tempo	Tow trains	Ruas	Meta-heurística
data[M1] = [(0, 50), # depot (10, 30), # Estação 1 (75, 85), # Estação 2 (15, 60), # Estação 3 (0, 40), # Estação 4 (30, 250), # Estação 5 (80, 95), # Estação 6 (5, 120), # Estação 7 (5, 120), # Estação 8 (5, 120), # Estação 9 (0, 60), # Estação 10 (0, 50), # Estação 11 (0, 40), # Estação 12 (5, 60), # Estação 13 (0, 80), # Estação 14 (0, 70), # Estação 15 (1, 80) # Estação 16]	8	Mão livre / Restrições no *line supply*	*Simulated Annealing*	data[M2] = [(0, 45), # depot (10, 25), # Estação 1 (75, 80), # Estação 2 (15, 55), # Estação 3 (0, 35), # Estação 4 (30, 245), # Estação 5 (80, 90), # Estação 6 (5, 115), # Estação 7 (5, 115), # Estação 8 (5, 115), # Estação 9 (0, 55), # Estação 10 (0, 50), # Estação 11 (0, 35), # Estação 12 (5, 55), # Estação 13 (0, 75), # Estação 14 (0, 65), # Estação 15 (1, 75) # Estação 16]	8	Mão livre / Restrições no *line supply*	*Simulated Annealing*
	7	Mão livre / Restrições no *line supply*	*Simulated Annealing*		7	Mão livre / Restrições no *line supply*	*Simulated Annealing*
	6	Mão livre / Restrições no *line supply*	*Simulated Annealing*		6	Mão livre / Restrições no *line supply*	*Simulated Annealing*
	5	Mão livre / Restrições no *line supply*	*Simulated Annealing*		5	Mão livre / Restrições no *line supply*	*Simulated Annealing*
	4	Mão livre / Restrições no *line supply*	*Simulated Annealing*		4	Mão livre / Restrições no *line supply*	*Simulated Annealing*
Janelas de tempo	**Tow trains**	**Ruas**	**Meta-heurística**	**Janelas de tempo**	**Tow trains**	**Ruas**	**Meta-heurística**
data[M1] = [(0, 50), # depot (10, 30), # Estação 1 (75, 85), # Estação 2 (15, 60), # Estação 3 (0, 40), # Estação 4 (30, 250), # Estação 5 (80, 95), # Estação 6 (5, 120), # Estação 7 (5, 120), # Estação 8 (5, 120), # Estação 9 (0, 60), # Estação 10 (0, 50), # Estação 11 (0, 40), # Estação 12 (5, 60), # Estação 13 (0, 80), # Estação 14 (0, 70), # Estação 15 (1, 80) # Estação 16]	8	Mão livre / Restrições no *line supply*	Vizinho Mais Próximo	data[M2] = [(0, 45), # depot (10, 25), # Estação 1 (75, 80), # Estação 2 (15, 55), # Estação 3 (0, 35), # Estação 4 (30, 245), # Estação 5 (80, 90), # Estação 6 (5, 115), # Estação 7 (5, 115), # Estação 8 (5, 115), # Estação 9 (0, 55), # Estação 10 (0, 50), # Estação 11 (0, 35), # Estação 12 (5, 55), # Estação 13 (0, 75), # Estação 14 (0, 65), # Estação 15 (1, 75) # Estação 16]	8	Mão livre / Restrições no *line supply*	Vizinho Mais Próximo
	7	Mão livre / Restrições no *line supply*	Vizinho Mais Próximo		7	Mão livre / Restrições no *line supply*	Vizinho Mais Próximo
	6	Mão livre / Restrições no *line supply*	Vizinho Mais Próximo		6	Mão livre / Restrições no *line supply*	Vizinho Mais Próximo
	5	Mão livre / Restrições no *line supply*	Vizinho Mais Próximo		5	Mão livre / Restrições no *line supply*	Vizinho Mais Próximo
	4	Mão livre / Restrições no *line supply*	Vizinho Mais Próximo		4	Mão livre / Restrições no *line supply*	Vizinho Mais Próximo

Fonte: Elaboração própria (2023).

Tabela 9 – Tabela de comparação 2

Janelas de tempo	Tow trains	Ruas	Meta-heurística	Janelas de tempo	Tow trains	Ruas	Meta-heurística
data[M3] = [(0, 35), (10, 15), (65, 70), (15, 45), (0, 25), (30, 235), (70, 80), (5, 105), (5, 105), (5, 105), (0, 45), (0, 35), (0, 25), (5, 45), (0, 65), (0, 55), (1, 65),]	8	Mão livre / Restrições no *line supply*	*Simulated Annealing*	data[M4] = [(0, 50), (10, 40), (75, 95), (15, 70), (0, 40), (30, 260), (80, 105), (5, 130), (5, 130), (5, 130), (0, 70), (0, 60), (0, 50), (5, 70), (0, 90), (0, 80), (1, 90),]	8	Mão livre / Restrições no *line supply*	*Simulated Annealing*
	7	Mão livre / Restrições no *line supply*	*Simulated Annealing*		7	Mão livre / Restrições no *line supply*	*Simulated Annealing*
	6	Mão livre / Restrições no *line supply*	*Simulated Annealing*		6	Mão livre / Restrições no *line supply*	*Simulated Annealing*
	5	Mão livre / Restrições no *line supply*	*Simulated Annealing*		5	Mão livre / Restrições no *line supply*	*Simulated Annealing*
	4	Mão livre / Restrições no *line supply*	*Simulated Annealing*		4	Mão livre / Restrições no *line supply*	*Simulated Annealing*
Janelas de tempo	**Tow trains**	**Ruas**	**Meta-heurística**	**Janelas de tempo**	**Tow trains**	**Ruas**	**Meta-heurística**
data[M3] = [(0, 35), (10, 15), (65, 70), (15, 45), (0, 25), (30, 235), (70, 80), (5, 105), (5, 105), (5, 105), (0, 45), (0, 35), (0, 25), (5, 45), (0, 65), (0, 55), (1, 65),]	8	Mão livre / Restrições no *line supply*	Vizinho Mais Próximo	data[M4] = [(0, 50), (10, 40), (75, 95), (15, 70), (0, 40), (30, 260), (80, 105), (5, 130), (5, 130), (5, 130), (0, 70), (0, 60), (0, 50), (5, 70), (0, 90), (0, 80), (1, 90),]	8	Mão livre / Restrições no *line supply*	Vizinho Mais Próximo
	7	Mão livre / Restrições no *line supply*	Vizinho Mais Próximo		7	Mão livre / Restrições no *line supply*	Vizinho Mais Próximo
	6	Mão livre / Restrições no *line supply*	Vizinho Mais Próximo		6	Mão livre / Restrições no *line supply*	Vizinho Mais Próximo
	5	Mão livre / Restrições no *line supply*	Vizinho Mais Próximo		5	Mão livre / Restrições no *line supply*	Vizinho Mais Próximo
	4	Mão livre / Restrições no *line supply*	Vizinho Mais Próximo		4	Mão livre / Restrições no *line supply*	Vizinho Mais Próximo
Janelas de tempo	**Tow trains**	**Ruas**	**Meta-heurística**	**Janelas de tempo**	**Tow trains**	**Ruas**	**Meta-heurística**
data[M5] = [(0, 100), (10, 80), (75, 135), (15, 110), (0, 90), (30, 300), (80, 145), (5, 170), (5, 170), (5, 170), (0, 110), (0, 100), (0, 90), (5, 110), (0, 130), (0, 120), (1, 130),]	8	Mão livre / Restrições no *line supply*	*Simulated Annealing*	data[M5] = [(0, 100), (10, 80), (75, 135), (15, 110), (0, 90), (30, 300), (80, 145), (5, 170), (5, 170), (5, 170), (0, 110), (0, 100), (0, 90), (5, 110), (0, 130), (0, 120), (1, 130),]	8	Mão livre / Restrições no *line supply*	Vizinho Mais Próximo
	7	Mão livre / Restrições no *line supply*	*Simulated Annealing*		7	Mão livre / Restrições no *line supply*	Vizinho Mais Próximo
	6	Mão livre / Restrições no *line supply*	*Simulated Annealing*		6	Mão livre / Restrições no *line supply*	Vizinho Mais Próximo
	5	Mão livre / Restrições no *line supply*	*Simulated Annealing*		5	Mão livre / Restrições no *line supply*	Vizinho Mais Próximo
	4	Mão livre / Restrições no *line supply*	*Simulated Annealing*		4	Mão livre / Restrições no *line supply*	Vizinho Mais Próximo

Fonte: Elaboração própria (2023).

Aponta-se que, nessas Tabelas, a fim de facilitar a posterior comparação de cenários, definiu-se que as matrizes de janelas de tempo seriam denominadas como M1, M2, M3, M4 e M5, respectivamente.

Destaca-se, ainda, que variando-se as duas meta-heurísticas e os *inputs* atrelados aos cenários definidos, totalizam-se 120 casos gerados e analisados. Em virtude de tal número elevado de simulações realizadas, foram priorizadas algumas análises com relação ao desempenho da otimização obtida pela ferramenta proposta. São elas:

- Análise do percentual de diferença de valores da distância máxima percorrida;
- Análise do percentual de diferença do número de *tow trains* utilizados e;
- Verificação da capacidade de atendimento das janelas de tempo.

Assim, torna-se possível sistematizar o processo de análise dos resultados, o qual será apresentado em sequência.

4.1.2 Análise dos resultados

Aqui serão abordados os resultados referentes ao desempenho do algoritmo em realizar a roteirização e o *schedule* de *tow trains*, incluindo os diversos cenários simulados e as restrições e condicionais próprias do problema em questão. Esses resultados podem ser classificados em resultados representativos de operações de otimização e resultados que indicam sucesso ou falha da operação, sendo que estes segundos tendem a atuar como parâmetros indicativos dos limites operacionais para o funcionamento das linhas de montagem simuladas.

Desse modo, analisando os resultados representativos da otimização, pode-se observar, em operações de roteirização bem-sucedidas, os níveis nos quais a operação foi otimizada. Ou seja, esses resultados destacam como diferentes valores dos *inputs* influenciam positivamente a eficiência da operação, seja reduzindo a distância total percorrida, minimizando o tempo de transporte ou otimizando outros critérios predefinidos.

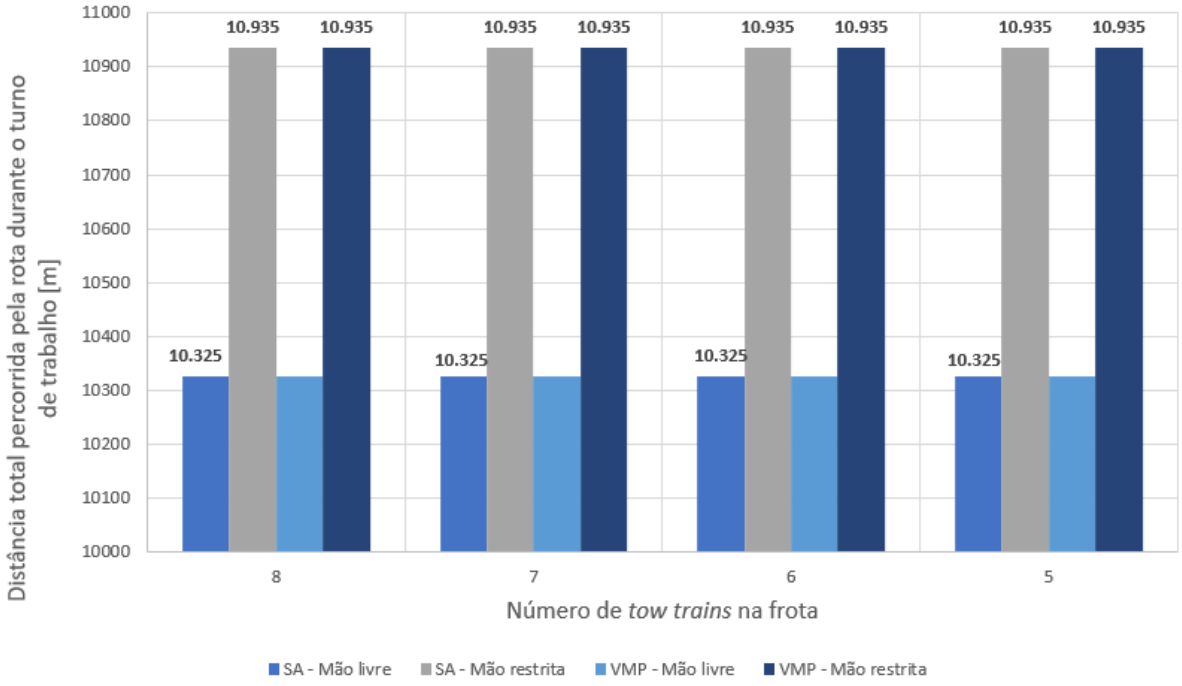
Já ao examinar os resultados de sucesso ou falha da operação, identificam-se os parâmetros que levam a falhas no reabastecimento da linha de montagem, delineando os limites em que a operação deixa de ser satisfatória (momento em que a demanda não pode ser atendida corretamente). Com isso, mostra-se necessário compreender esses limites para avaliar as condições em que o processo de roteirização pode falhar em atingir seus objetivos.

4.1.2.1 Diminuição da frota de tow trains

Em relação à redução da frota de *tow trains*, destaca-se que esse é um dos principais pontos de consideração na otimização do processo de roteirização e *schedule*

das linhas de montagem, uma vez que esta possui potencial para gerar economias significativas, reduzindo custos operacionais, tais como manutenção, combustível e mão de obra. Vale notar, no entanto, que a redução da frota não impacta diretamente na otimização das distâncias percorridas, como apresentado na Figura 33, uma vez que os *tow trains* não necessários permanecem no supermercado e a redução extrema implica no não atendimento da demanda.

Figura 33 – Frota de *tow trains* e distâncias percorridas



Fonte: Elaboração própria (2023).

Ao examinar detalhadamente o gráfico da Figura 33, torna-se evidente a estabilização das distâncias totais, mesmo diante de flutuações no número de veículos presentes na frota. Esse comportamento reflete a capacidade do algoritmo em atingir um ponto ótimo entre o número de veículos na frota e as distâncias percorridas, uma vez que os *tow trains* não utilizados não executam rotas desnecessárias, otimizando assim o uso dos recursos disponíveis. Especificamente, o algoritmo alcançou valores ótimos de distâncias totais percorridas nas rotas durante um turno de trabalho iguais a: 10.325 metros nos cenários sem restrições no *line supply* e 10.935 metros nos cenários com restrições.

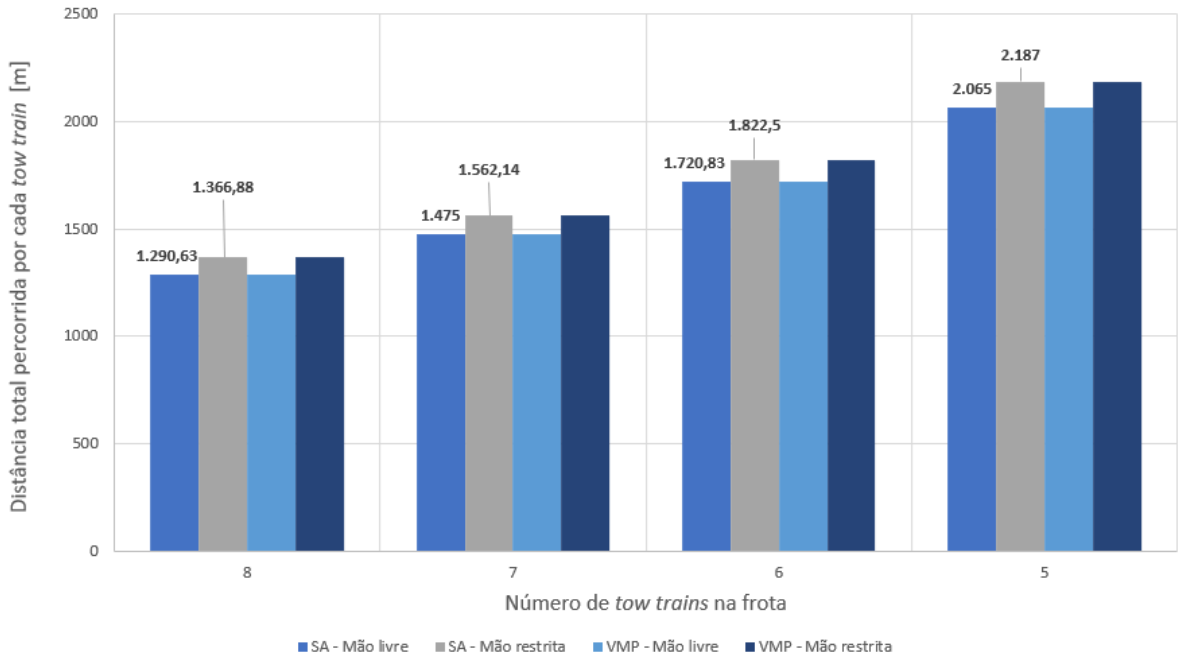
Após a realização dos testes em diversos cenários com *inputs* diversos, é notável que tais valores de distâncias, foram consistentemente apresentados pelo algoritmo como os mais otimizados. Essa consistência ao longo de diferentes condições reforça a confiança de que esses valores representam a solução ótima para a estrutura

da montadora simulada, validando a eficiência e estabilidade do algoritmo.

Nesse contexto analisado, é válido mencionar o ponto ótimo está associado com a configuração em que o menor número possível de *tow trains* é capaz de percorrer a distância ótima. Em outras palavras, o algoritmo busca otimizar não apenas a distância percorrida, mas também a frota de veículos, ou seja, maximizar a eficiência individual de cada *tow train*. Portanto, o ponto ótimo é caracterizado pelo cenário em que a menor quantidade de *tow trains* é capaz de cobrir a distância total ótima, atendendo completamente a demanda, maximizando a eficiência dos veículos e assegurando a máxima utilização dos recursos disponíveis.

Com isso, mostra-se que a frota com 5 *tow trains* apresenta vantagens frente às demais, uma vez que a demanda é atendida corretamente e as distâncias ótimas são atingidas (10.325 e 10.935 metros). Tal comentário pode ser verificado na Figura 34 em sequência, a qual demonstra as distâncias percorridas por cada *tow train* durante um turno de trabalho da montadora.

Figura 34 – Distâncias totais percorridas por cada *tow train*



Fonte: Elaboração própria (2023).

Nessa Figura 34 evidencia-se que a frota de *tow trains* com 5 veículos apresentou valores de distâncias totais percorridas por veículo iguais a 2.065 e 2.187 metros nos cenários com mão livre e mão restrita, respectivamente. Ao comparar-se tais resultados com as distâncias percorridas pelos cenários com frotas mais extensas (8, 7 e 6 *tow trains*), demonstra-se a máxima utilização de recursos na frota reduzida (entre 37,5% e 16,67%), comprovando-se o ponto ótimo de operação (Tabela 10).

Tabela 10 – Comparação entre ponto ótimo e demais pontos

Número de *tow trains*	**Resultado mão livre**	**Diferença absoluta (N5 - N8)**	**Diferença percentual (N5 - N8)**
8	1.290,63	774,38	37,50%
7	1.475,00	590,00	28,57%
6	1.720,83	344,17	16,67%
Número de *tow trains*	**Resultado mão restrita**	**Diferença absoluta (N5 - N8)**	**Diferença percentual (N5 - N8)**
8	1.366,875	820,13	37,50%
7	1.562,1429	624,86	28,57%
6	1.822,5	364,50	16,67%

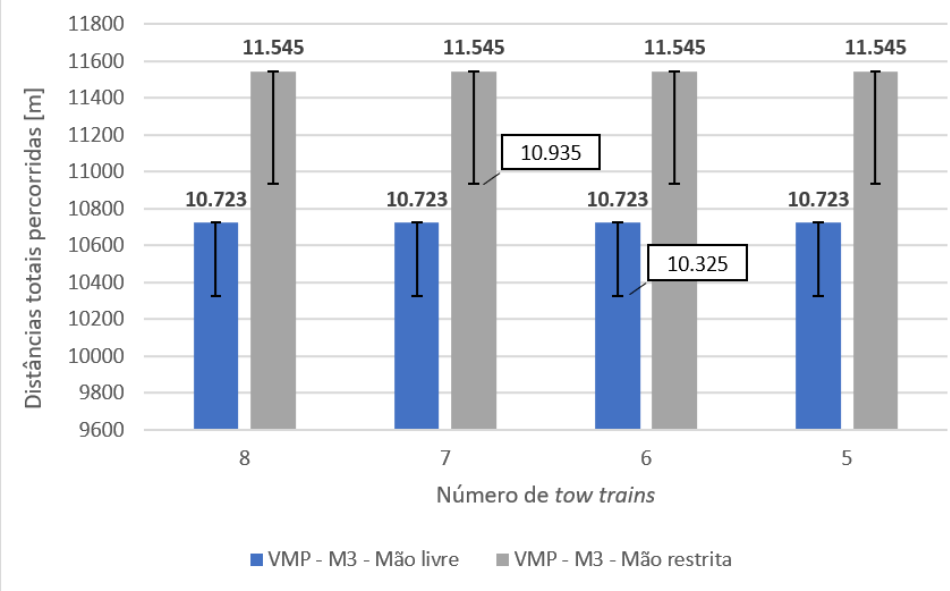
Fonte: Elaboração própria (2023).

Adicionalmente, é importante notar que os cenários com 4 e 3 *tow trains* na frota não estão incluídos nos resultados comparativos de distâncias, uma vez que apresentaram inconsistências no atendimento da demanda. A inclusão desses cenários poderia levar a resultados distorcidos e, portanto, não foram considerados na análise.

Outro ponto de destaque na análise dos resultados de redução da frota é que, dentre todos os cenários analisados, apenas dois não obtiveram uma distância total percorrida equivalente aos valores ótimos (10.325 e 10.935 metros): os cenários simulados no algoritmo com a heurística do Vizinho Mais Próximo, correspondentes à matriz de janelas de tempos M3, com e sem restrições no *line supply*. Esses cenários podem ser visualizados na Figura 35, a qual demonstra a diferença absoluta entre os valores obtidos nesse cenário e os valores ótimos anteriormente mencionados.

Esses resultados podem ser atribuídos à combinação das restrições de frota com as restrições de janela de tempo, tornando o cenário de análise complexo. Nesse contexto, o método do Vizinho Mais Próximo, que possui um horizonte de busca reduzido em comparação com o *Simulated Annealing*, retornou um resultado sub-ótimo devido à dificuldade de englobar todas as restrições em uma única solução.

Figura 35 – Frota de *tow trains* e distâncias sub-ótimas



Fonte: Elaboração própria (2023).

Conforme indicado na Tabela 11, o cenário sem limitações no *line supply* apresenta um erro relativo de 3,85% em relação à solução ótima, enquanto o cenário com limitações de direção possui um erro de 5,58%.

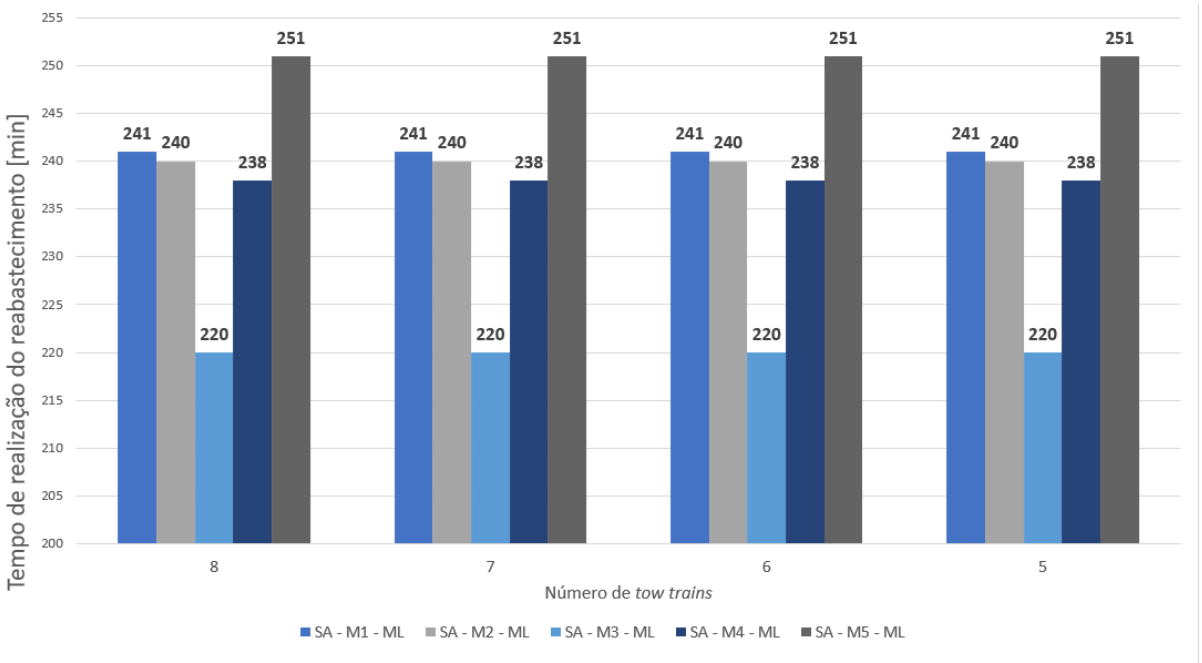
Tabela 11 – Diferenças percentuais entre cenários com e sem restrições no *line supply*

Resultado mão restrita ótimo	**Resultado mão restrita sub-ótimo**	**Diferença percentual**
10.935	11.545	3,85%
Resultado mão livre ótimo	**Resultado mão livre sub-ótimo**	**Diferença percentual**
10.325	10.723	5,58%

Fonte: Elaboração própria (2023).

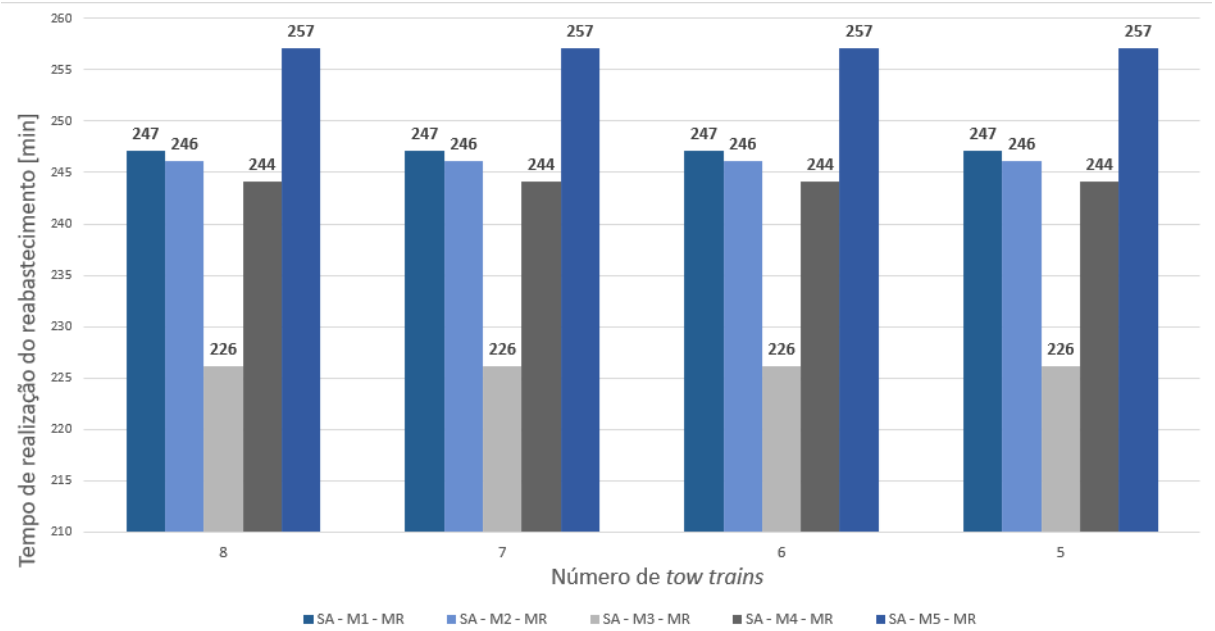
No que tange ao tempo de realização do processo de reabastecimento das linhas, verificou-se que a alteração do número de veículos na frota de *tow trains* não possui influência direta para o parâmetro, como apresentado nas Figuras 36 e 37.

Figura 36 – Frota de *tow trains* e tempos de rotas em cenários sem restrições no *line supply*



Fonte: Elaboração própria (2023).

Figura 37 – Frota de *tow trains* e tempos de rotas em cenários com restrições no *line supply*



Fonte: Elaboração própria (2023).

Ao analisar as Figuras 36 e 37 que indicam a relação entre a redução da frota de *tow trains* e tempo de realização do reabastecimento da linha, percebe-se

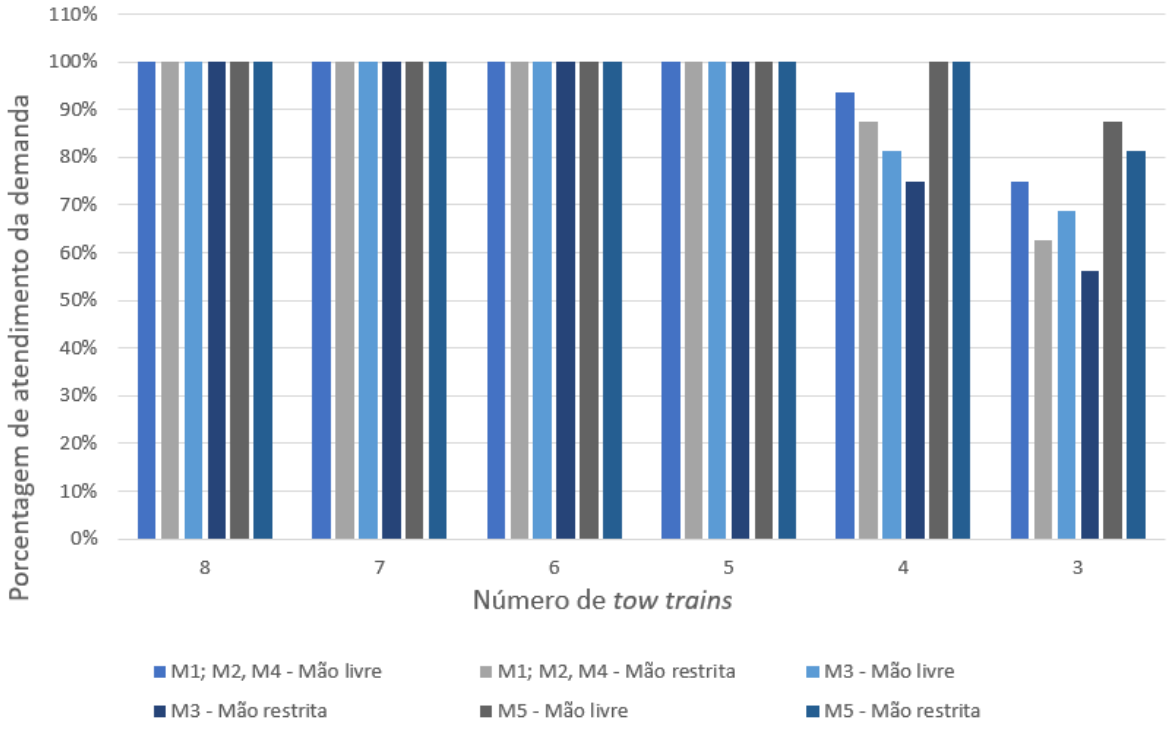
um caráter similar entre os cenários simulados. Isso ocorre, pois os parâmetros de restrições de direções no *line supply* e janelas de tempo têm um efeito dominante nessas métricas. Desse modo, visualiza-se que o tempo empregado para realizar uma rota completa de reabastecimento permanece constante nos cenários com mesmas janelas de tempo e restrições do *line supply* (por exemplo, para a matriz M1 o tempo de realização das rotas é igual a 247 minutos no cenário com restrições no *line supply* para as frotas de 8, 7, 6 e 5 *tow trains*), ainda que o número de veículos na frota tenha sido alterado - como se verifica nos gráficos 36 e 37 a partir da repetição de tempos iguais com números de *tow trains* diferentes (8, 7, 6 e 5):

- Cenário com matriz de janelas de tempos M1 e mão livre: 241 minutos
- Cenário com matriz de janelas de tempos M2 e mão livre: 240 minutos
- Cenário com matriz de janelas de tempos M3 e mão livre: 220 minutos
- Cenário com matriz de janelas de tempos M4 e mão livre: 238 minutos
- Cenário com matriz de janelas de tempos M4 e mão livre: 251 minutos
- Cenário com matriz de janelas de tempos M1 e mão restrita: 247 minutos
- Cenário com matriz de janelas de tempos M2 e mão restrita: 246 minutos
- Cenário com matriz de janelas de tempos M3 e mão restrita: 226 minutos
- Cenário com matriz de janelas de tempos M4 e mão restrita: 244 minutos
- Cenário com matriz de janelas de tempos M4 e mão restrita: 257 minutos

Em vez disso, a redução da frota afeta a capacidade de atendimento da demanda, uma vez que se não houver veículos suficientes para atender todas as estações de trabalho dentro dos limites impostos, a operação de roteirização falhará. Com isso, verificou-se que essa redução não pode ser realizada de maneira excessiva (abaixo de 5 *tow trains* no presente estudo), em virtude da existência de um *trade off* intrínseco entre a diminuição da frota e a capacidade de atender a demanda.

Nas presentes simulações, considerando-se que a demanda permanece constante entre os diferentes cenários, verificou-se que a redução de *tow trains* e o atendimento da demanda possuem uma relação na forma de uma curva decrescente, como apresentado na Figura 38. Isso indica que a redução do número de veículos não equivale necessariamente a uma operação mais eficiente, uma vez que precisa-se considerar o cumprimento eficaz da demanda.

Figura 38 – Frota de *tow trains* e atendimento da demanda



Fonte: Elaboração própria (2023).

Como evidenciado na Figura 38, à medida que o se reduz o número de *tow trains* presentes na frota o atendimento da demanda começa a apresentar falhas (atendendo em média entre 53,25% a 9,75% da demanda), as quais são indicadas pelas estações ignoradas durante o reabastecimento da linha de montagem. Durante a execução do algoritmo, ainda, verificou-se que o número mínimo de *tow trains* para o atendimento das demandas aplicadas foi 4, uma vez que nenhum cenário com 3 veículos apresentou sucesso nas visitas de todas as estações.

A Tabela 12 em sequência evidencia os percentuais de demandas atendidas nos cenários com falhas no atendimento das estações da linha. Nessa, comprova-se a afirmação de que a redução de *tow trains* na frota tende a aumentar o percentual de falha no atendimento da demanda.

Tabela 12 – Percentual de atendimento da demanda

Cenários	Número de *tow trains*	Percentual de atendimento da demanda
M1; M2, M4 - Mão livre	4	93,75%
M1; M2, M4 - Mão restrita		87,50%
M3 - Mão livre		81,25%
M3 - Mão restrita		75,00%
M5 - Mão livre		100%
M5 - Mão restrita		100%
Cenários	**Número de *tow trains***	**Percentual de atendimento da demanda**
M1; M2, M4 - Mão livre	3	75,00%
M1; M2, M4 - Mão restrita		62,50%
M3 - Mão livre		68,75%
M3 - Mão restrita		56,25%
M5 - Mão livre		87,50%
M5 - Mão restrita		81,25%

Fonte: Elaboração própria (2023).

Outro resultado relevante é a interação entre a redução da frota de *tow trains* e as janelas de tempo. Os resultados revelam que a redução da frota pode ser mais prejudicial para o atendimento da demanda nas situações em que as janelas de tempo são menores (cerca de 18,75%, por exemplo, ao comparar-se o cenário da matriz mais restrita, M3, com a matriz mais flexível, M5).

Verificou-se que nos cenários com janelas de tempo mais curtas (M4 e M3), a eliminação de veículos adquiriu um impacto negativo na capacidade de cumprir as demandas da linha de montagem (ou seja, o número de estações não atendidas aumentou), uma vez que o tempo necessário para um veículo atender múltiplas estações se reduz proporcionalmente à restrição das janelas de tempo (por exemplo, ao comparar a matriz M4 com a matriz M3, percebe-se que os *tow trains* terão o tempo para atendimento de cada estação reduzido em 10 minutos no cenário da janela M3), porém sua velocidade de atendimento permanece constante (entre 0 a 6km/h).

Esse comentário é comprovado ao analisar a Figura 38 e a Tabela 12, uma vez que nestes evidencia-se que o cenário com a matriz de janelas de tempo mais restrita, M3, apresentou a menor taxa de atendimento de demanda, já o cenário com a matriz mais flexível, M5, apresentou as melhores taxas:

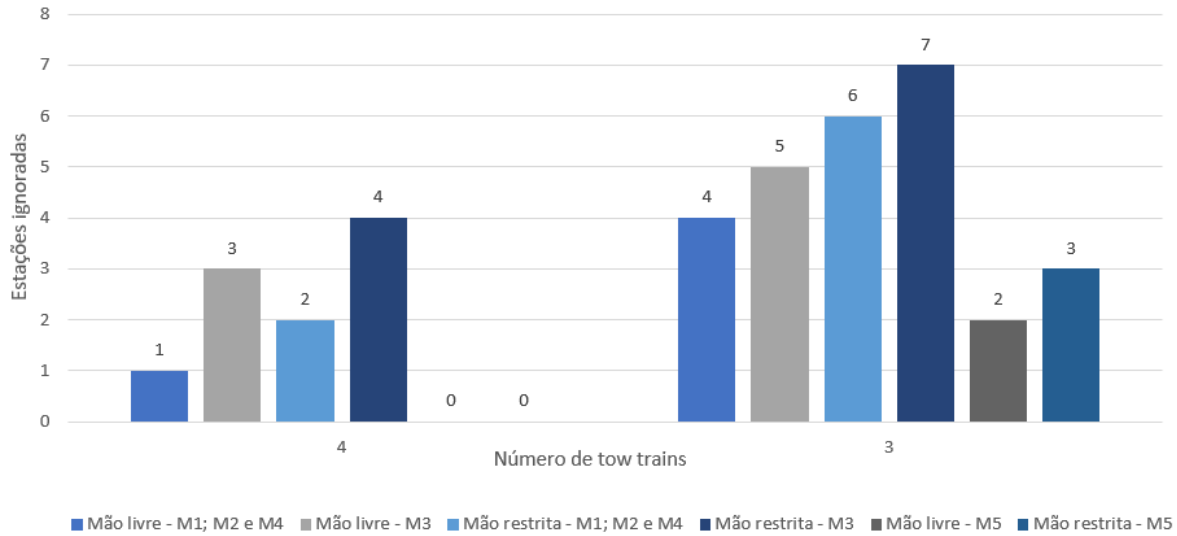
- M3 - Mão restrita: taxa de 75% e 56,25% de atendimento de demanda para 4 e 3 *tow trains*, respectivamente.
- M3 - Mão livre: taxa de 81,25% e 68,75% de atendimento de demanda para 4 e 3 *tow trains*, respectivamente

- M5 - Mão restrita: taxa de 100% e 81,25% de atendimento de demanda para 4 e 3 *tow trains*, respectivamente.
- M5 - Mão livre: taxa de 100% e 87,50% de atendimento de demanda para 4 e 3 *tow trains*, respectivamente

De maneira semelhante, a Figura 39 representa a relação negativa entre a redução da frota de veículos e o atendimento adequado das janelas de tempo (cenários em que os veículos visitaram as estações nos tempos correspondentes às janelas de tempo e, assim, conseguiram atender às demandas das estações), como verifica-se ao analisar o cenário com frota de *tow trains* igual a 3 veículos, o qual apresenta maiores números de estações não atendidas em comparação com o cenário com frota igual a 4 veículos (diferença percentual entre 40% e 66,6%).

Paralelamente, a Figura 39 possibilita verificar, a flexibilidade que janelas de tempo mais amplas fornecem para a roteirização de veículos, uma vez que o cenário da matriz M5 (matriz mais flexível) forneceu os melhores resultados no que tange ao atendimento eficiente das demandas (0 estações não atendidas no cenário de frota de *tow trains* igual a 4 veículos e 2 estações no cenário com 3 *tow trains*).

Figura 39 – Frota de *tow trains* e estações não atendidas



Fonte: Elaboração própria (2023).

Ao analisar a Figura 39 e a Tabela 12, reitera-se que a redução da frota de *tow trains* possui uma importância considerável no atendimento da demanda (podendo representar uma queda de até 43,75% no atendimento de estações), indicando um *trade-off* entre este fator e a otimização do processo de reabastecimento. Ainda, de

maneira integrada, os parâmetros de controle de janelas de tempo e restrições no *line supply* relacionam-se com a frota para garantir o atendimento da demanda, porém, esses primeiros, possuem maior destaque na otimização de tempos e distâncias.

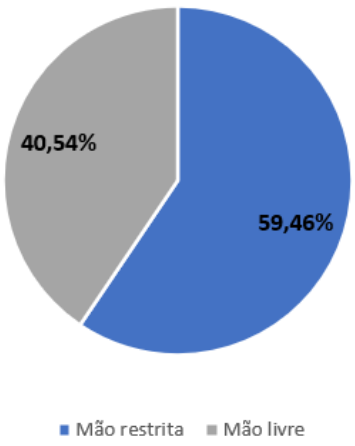
4.1.2.2 Tornar todas as ruas com idas e vindas

Ainda, evidenciam-se os resultados gerados a partir da relação entre a redução do número de veículos e as limitações de direção nas ruas do *line supply*, as quais determinam as rotas pelas quais os *tow trains* podem se movimentar. Ao executar o código, verificou-se que a redução da frota torna-se crítica em cenários com limitações direções, uma vez que é comum que os veículos tenham que seguir rotas alternativas, por vezes mais longas e demoradas, para atender a uma estação de trabalho específica devido a essas restrições.

Isso significa que, em termos de otimização, a eficiência é comprometida, uma vez que as rotas não podem ser escolhidas puramente com base na proximidade geográfica e no atendimento de janelas de tempo. Nesse cenário, a redução da frota de *tow trains* torna-se ainda mais crítica para o atendimento da demanda. Afinal, quando as rotas já são complexas devido às restrições de direções, a eliminação de veículos gera uma pressão adicional, a qual soma-se às restrições de janela de tempo, culminando em estações não atendidas.

A Figura 40, em paralelo com a Figura 39, demonstra como os cenários com ruas delimitadas tendem a acumular estações ignoradas durante o reabastecimento, enquanto os cenários com ruas de mão livre permitem uma redução maior da frota sem apresentar problemáticas para o atendimento da demanda.

Figura 40 – Restrições no *line supply* e estações não atendidas



Fonte: Elaboração própria (2023).

Verificam-se os cenários com restrições no *line supply* apresentando 59,46% de todas as estações não atendidas, indicando a relação negativa entre limitação de direções e atendimento da demanda.

Ao analisar os resultados atrelados ao número de *tow trains* integrantes da frota, verifica-se que estes adotam tanto um caráter de indicador do desempenho da otimização (uma vez que indicam os pontos ótimos na relação entre distâncias percorridas e tamanho da frota de *tow trains*), quanto de sucesso ou falha da operação (levando-se em consideração que os resultados com frotas reduzidas - 4 e 3 veículos - demonstraram falhas no atendimento da demanda, ou seja, falhas no processo de roteirização). Como pode-se verificar nas Figuras 39 e 40, dentre os cenários que apresentam falha no atendimento da demanda (todos os cenários com frota de *tow trains* igual a 3 e os cenários das matrizes de janelas de tempo M1, M2, M3 e M4 com número de *tow trains* igual a 4), os casos com direções restritas demonstram a maior ocorrência de estações com demandas não atendidas (59,46%), justificando o caráter de falha na operação.

Por outro lado, analisando-se os gráficos contidos nas Figuras 36 e 33 verifica-se que:

- os cenários com restrições no *line supply* apresentam um aumento de 5.91% nas distâncias totais percorridas
- os cenários com restrições no *line supply* apresentam um aumento de cerca de 6 minutos no tempo total de reabastecimento da linha de montagem

Desse modo, revela-se que, em termos de otimização da operação, a restrição de rotas não apresenta vantagens. As vantagens de tal abordagem ligam-se com a simplificação da operação e da manobrabilidade dos *tow trains*, uma vez que, ao limitar as rotas possíveis, o planejamento e a execução das viagens dos *tow trains* tornam-se mais diretos e previsíveis.

4.1.2.3 Aumentar e reduzir as janelas de tempo

Quanto aos resultados ligados às alterações das janelas de tempo (aumento e redução), verifica-se que a princípio, a redução das janelas de tempo é considerada uma estratégia eficaz para otimizar o tempo total destinado para o reabastecimento da linha de montagem. No entanto, semelhante ao apresentado com a redução da frota de *tow trains*, uma janela de tempo excessivamente exígua pode resultar em problemas substanciais no que tange ao atendimento pontual das estações de trabalho. A restrição excessiva das janelas de tempo tem o potencial de criar limitações rígidas que inviabilizam a entrega no prazo, prejudicando tanto a produtividade quanto a eficiência operacional.

Esse *trade off* pode ser verificado na Figura 38, a qual demonstra a curva

decrescente entre o atendimento da demanda e a diminuição dos tempos permitidos de visita. De maneira contrária, verifica-se nessa que o aumento das janelas de tempo permite que a demanda seja atendida com uma taxa de sucesso superior

Esse comentário é verificado ao analisar-se que a matriz de janelas de tempo M5, a qual possui os maiores intervalos de visitas, apresenta os menores números de estações não atendidas (2 estações no cenário com frota de *tow trains* igual a 3 veículos e sem restrição de direção no *line supply* e 3 estações no cenário com frota de *tow trains* igual a 3 veículos e com restrição *line supply*).

Nesse contexto, aumentar o intervalo de tempo alocado para cada entrega mostra-se como uma alternativa prudente para garantir o funcionamento da linha, uma vez que janelas de tempo mais amplas conferem maior flexibilidade para lidar com eventualidades e flutuações na demanda (no presente estudo, tal flexibilidade representa um aumento entre 12,5% e 18,75% no atendimento da demanda em comparação com o cenário de janela de tempo mais restrita, M3). Todavia, outro *trade off* traduz-se em um aumento do tempo total despendido na entrega.

Ainda, verifica-se que a flexibilidade conferida pelo aumento de janelas de tempo tem um impacto considerável no tempo total de reabastecimento da linha de montagem, como observado nas Figuras 36 e 37 as quais demonstram que os cenários com a matriz de tempos mais flexível (M5) apresentam um aumento de 31 minutos no tempo de realização do reabastecimento em comparação com a matriz mais restrita (M3). Isso ocorre, pois janelas de tempo mais amplas permitem aos *tow trains* acomodar múltiplas estações de trabalho em uma única viagem, permitindo a possível seleção de estações de trabalho em sequência, o que equivale a uma rota mais direta e eficiente.

Como o algoritmo aqui desenvolvido busca otimizar as distâncias percorridas e o número de *tow trains*, e não o tempo de realização das rotas, verifica-se que a adoção de janelas de tempo mais amplas favorecem a otimização do número de *tow trains* e distâncias percorridas, contudo, esse processo pode implicar no aumento do tempo de realização das entregas. Isso ocorre em decorrência do acúmulo de entregas por um único veículo, característica favorecida por uma janela de tempo ampla, a qual ocorre em função dos parâmetros de otimização escolhidos.

A Tabela 13 demonstra tal comentário ao comparar os resultados dos cenários simulados com as matrizes de janelas de tempo mais e menos restritas (M3 e M5, respectivamente). Nessa, visualiza-se que o algoritmo busca alcançar a mínima distância percorrida, obtendo sucesso no atendimento total da demanda para os cenários com ambas matrizes de janelas de tempo em frotas de 8 a 5 *tow trains*. O cenário com a matriz flexível, no entanto, apresentou sucesso no atendimento das estações no cenário com a frota de veículos igual 4, enquanto o cenário da matriz restrita apresentou falha (apenas 81,25% da demanda foi atendida).

Ao analisar-se o tempo de realização do processo de roteirização, no entanto,

verifica-se que a matriz flexível apresenta maior tempo despendido para a realização de entregas (31 minutos), revelando o *trade off*.

Tabela 13 – Comparação de janelas de tempos e cenários simulados

Janela de tempo	**Número de *tow trains***	**% atendimento da demanda**	**Tempo de realização de rota**	**Distâncias percorridas**
M5 (cenário sem restrições no *line supply*)	8	100%	251	10.325
	7	100%	251	10.325
	6	100%	251	10.325
	5	100%	251	10.325
	4	100%	251	-
	3	87,5%	251	-
Janela de tempo	**Número de *tow trains***	**% atendimento da demanda**	**Tempo de realização de rota**	**Distâncias percorridas**
M3 (cenário sem restrições no *line supply*)	8	100%	220	10.325
	7	100%	220	10.325
	6	100%	220	10.325
	5	100%	220	10.325
	4	81,25%	220	-
	3	68,75%	220	-

Fonte: Elaboração própria (2023).

Com a análise da Tabela 13 e das Figuras 36 e 37, então, demonstra-se essa curva positiva entre o aumento das janelas de tempo e o aumento do tempo de realização de entregas. O contrário demonstra-se quando as janelas de tempo são mais restritas. Com intervalos de tempo mais curtos, os *tow trains* precisam garantir o atendimento de cada estação de trabalho estritamente dentro desses limites, o que pode levar a viagens mais curtas, mas mais frequentes (com tempos entre 2 a 31 minutos mais reduzidos para os cenários simulados):

- A matriz de janelas de tempo M3 (matriz mais restrita) apresenta os menores tempos de realização de rotas: 220 e 226 minutos com mãos livre e restritas, respectivamente.
- A matriz de janelas de tempo M5 (matriz mais ampla) apresenta os maiores tempos de realização de rotas: 251 e 257 minutos com mãos livre e restritas, respectivamente.

O gerenciamento das janelas de tempo no panorama das operações de uma linha de montagem deve ser meticulosamente ponderado, uma vez que pode diminuir a otimização das operações (Tabela 14).

Tabela 14 – Comparação de cenários simulados

Janela de tempo	**Número de *tow trains***	**% atendimento da demanda**	**Tempo de realização de rota**	**Distâncias percorridas**
M1 (cenário sem restrições no *line supply*)	8	100%	241	10.325
	7	100%	241	10.325
	6	100%	241	10.325
	5	100%	241	10.325
	4	93,75%	-	-
	3	75%	-	-
Janela de tempo	**Número de *tow trains***	**% atendimento da demanda**	**Tempo de realização de rota**	**Distâncias percorridas**
M2 (cenário sem restrições no *line supply*)	8	100%	240	10.325
	7	100%	240	10.325
	6	100%	240	10.325
	5	100%	240	10.325
	4	93,75%	-	-
	3	75%	-	-
Janela de tempo	**Número de *tow trains***	**% atendimento da demanda**	**Tempo de realização de rota**	**Distâncias percorridas**
M3 (cenário sem restrições no *line supply*)	8	100%	220	10.325
	7	100%	220	10.325
	6	100%	220	10.325
	5	100%	220	10.325
	4	81,25%	-	-
	3	68,75%	-	-
Janela de tempo	**Número de *tow trains***	**% atendimento da demanda**	**Tempo de realização de rota**	**Distâncias percorridas**
M4 (cenário sem restrições no *line supply*)	8	100%	238	10.325
	7	100%	238	10.325
	6	100%	238	10.325
	5	100%	238	10.325
	4	93,75%	-	-
	3	75%	-	-
Janela de tempo	**Número de *tow trains***	**% atendimento da demanda**	**Tempo de realização de rota**	**Distâncias percorridas**
M5 (cenário sem restrições no *line supply*)	8	100%	251	10.325
	7	100%	251	10.325
	6	100%	251	10.325
	5	100%	251	10.325
	4	100%	-	-
	3	87,5%	-	-

Fonte: Elaboração própria (2023).

Com a análise dessa Tabela, entre as janelas de tempo simuladas, infere-se que as janelas com aumento de 10 minutos (M4) são as mais vantajosas em virtude dos resultados positivos em otimização de distâncias e tempos, atendimento da demanda e número de *tow trains* utilizados (o cenário apresenta o segundo menor tempo de realização de rotas, dentre os demais - 238 minutos -, apresenta os segundos melhores níveis de atendimento da demanda - entre 75% e 93,75% - e alcança o valor de distância ótima percorrida - 10.325).

Os resultados referentes a esses cenários enquadram-se, assim, como atrelados ao processo de otimização e garantia de sucesso ou falha da operação.

4.2 ANÁLISE DE CUSTO COMPUTACIONAL DO ALGORITMO

De modo semelhante ao apresentando na comparação dos resultados atrelados aos distintos cenários simulados, a presente seção apoia-se na criação de uma tabela comparativa entre os resultados de custo computacional obtidos a partir da execução do algoritmo baseado na meta-heurística *simulated annealing* e, outro baseado no método do vizinho mais próximo.

Destaca-se que, a fim de testar os dois métodos, todos os cenários foram testados e tabelados, representando o ponto de partida para avaliar o desempenho dessas duas abordagens distintas. Ainda, levando-se em consideração que os resultados da otimização e da busca de soluções por um algoritmo dependem das características do *hardware* no qual esse é executado, aponta-se que o computador no qual o algoritmo foi implementado conta com as seguintes configurações:

- Notebook Acer Nitro 5;
- Processador Intel Core i5 (um processador de média eficiência que oferece um desempenho sólido e é capaz de lidar com tarefas computacionalmente intensivas, o que é fundamental ao avaliar o desempenho de algoritmos de otimização);
- SSD NVMe 250 GB (oferece tempos de leitura e gravação significativamente mais rápidos do que discos rígidos comuns).
- Memória de 8 GB (embora esta seja uma quantidade adequada para a maioria das tarefas, deve ser monitorada durante a execução do algoritmo, uma vez que o consumo de memória pode afetar o desempenho).
- Sistema operacional Windows 11 (possui um papel significativo nos resultados de desempenho, pois influencia o gerenciamento de recursos e a alocação de CPU e memória).

Em termos de tempo de execução, o método *Simulated Annealing* frequentemente exige mais iterações para convergir para uma solução, especialmente em problemas complexos. Por outro lado, o vizinho mais próximo tende a convergir mais rapidamente, o que reduz o tempo de execução do algoritmo. A Figura 41 demonstra a

comparação entre o tempo de computação dos dois solucionadores.

Figura 41 – Comparação de tempos de execução dos algoritmos

		Simulated Annealing	Vizinho Mais Próximo		Simulated Annealing	Vizinho Mais Próximo
Tow trains		Tempo de execução (s)	Tempo de execução (s)		Tempo de execução (s)	Tempo de execução (s)
8		29,95	29,83		30,25	30,14
7		30,11	29,84		30,41	30,23
6	M1 - Mão livre	30,21	29,93	M1 - Mão restrita	30,51	30,34
5		30,35	30,05		30,65	30,5
4		30,4	30,14		30,7	30,6
3		30,23	30,01		30,53	30,47
Tow trains		Tempo de execução (s)	Tempo de execução (s)		Tempo de execução (s)	Tempo de execução (s)
8		29,57	29,44		29,87	29,76
7		29,68	29,57		30,01	29,87
6	M2 - Mão livre	29,84	29,66	M2 - Mão restrita	30,17	30,01
5		29,93	29,82		30,35	30,1
4		30,12	29,95		30,4	30,19
3		29,88	29,86		30,3	30,13
Tow trains		Tempo de execução (s)	Tempo de execução (s)		Tempo de execução (s)	Tempo de execução (s)
8		30,05	29,93		30,35	30,25
7		30,14	30,04		30,5	30,36
6	M3 - Mão livre	30,26	30,13	M3 - Mão restrita	30,61	30,49
5		30,41	30,27		30,7	30,57
4		30,48	30,36		30,86	30,66
3		30,44	30,29		30,77	30,59
Tow trains		Tempo de execução (s)	Tempo de execução (s)		Tempo de execução (s)	Tempo de execução (s)
8		29,75	29,63		30,05	29,95
7		29,86	29,74		30,18	30,06
6	M4 - Mão livre	29,99	29,83	M4 - Mão restrita	30,32	30,16
5		30,08	29,97		30,4	30,25
4		30,12	30,06		30,54	30,36
3		30,11	30,02		30,45	30,29
Tow trains		Tempo de execução (s)	Tempo de execução (s)		Tempo de execução (s)	Tempo de execução (s)
8		29,72	29,59		30,02	29,91
7		29,83	29,7		30,17	30,07
6	M5 - Mão livre	29,92	29,83	M5 - Mão restrita	30,28	30,24
5		30,05	29,92		30,37	30,33
4		30,12	30,03		30,53	30,44
3		30,08	30		30,45	30,36

Fonte: Elaboração própria (2023).

Nessa Figura, é visível o maior tempo de computação agregado ao algoritmo com a meta-heurística *Simulated Annealing*. Ainda assim, as diferenças nos tempos encontram-se entre 0,067% e 0,988% e, portanto, não são um fator decisivo na escolha da heurística mais eficiente.

Outro resultado a ser analisado é a qualidade da solução, a qual desempenha um papel crucial na avaliação do custo computacional. O *Simulated Annealing*, por ser uma meta-heurística que explora soluções com base em probabilidades, tem uma vantagem potencial em encontrar soluções de maior qualidade quando comparado ao Vizinho Mais Próximo. Isso resulta em uma maior probabilidade de atingir aos objetivos definidos pelo algoritmo, encontrando uma solução ótima.

A Figura 42 demonstra a comparação entre os valores de distâncias máximas percorridas pelos *tow trains* nos algoritmos com os dois métodos de busca. Nessa, evidencia-se que, ainda que o método do Vizinho mais Próximo tenha apresentados resultados satisfatórios, este diferiu da solução ótima em dois cenários de análise: Matriz 3 com e sem restrições no *line supply*.

Figura 42 – Comparação de tempos de distâncias percorridas

		Simulated Annealing	Vizinho Mais Próximo		*Simulated Annealing*	Vizinho Mais Próximo
Tow trains		Distância total (m)	Distância total (m)		Distância total (m)	Distância total (m)
8		10.325	10.325		10.935	10.935
7		10.325	10.325		10.935	10.935
6	M1 - Mão livre	10.325	10.325	M1 - Mão restrita	10.935	10.935
5		10.325	10.325		10.935	10.935
4		-	-		-	-
3		-	-		-	-
Tow trains		Distância total (m)	Distância total (m)		Distância total (m)	Distância total (m)
8		10.325	10.325		10.935	10.935
7		10.325	10.325		10.935	10.935
6	M2 - Mão livre	10.325	10.325	M2 - Mão restrita	10.935	10.935
5		10.325	10.325		10.935	10.935
4		-	-		-	-
3		-	-		-	-
Tow trains		Distância total (m)	Distância total (m)		Distância total (m)	Distância total (m)
8		10.325	10.723		10.935	11.545
7		10.325	10.723		10.935	11.545
6	M3 - Mão livre	10.325	10.723	M3 - Mão restrita	10.935	11.545
5		10.325	10.723		10.935	11.545
4		-	-		-	-
3		-	-		-	-
Tow trains		Distância total (m)	Distância total (m)		Distância total (m)	Distância total (m)
8		10.325	10.325		10.935	10.935
7		10.325	10.325		10.935	10.935
6	M4 - Mão livre	10.325	10.325	M4 - Mão restrita	10.935	10.935
5		10.325	10.325		10.935	10.935
4		-	-		-	-
3		-	-		-	-
Tow trains		Distância total (m)	Distância total (m)		Distância total (m)	Distância total (m)
8		10.325	10.325		10.935	10.935
7		10.325	10.325		10.935	10.935
6	M5 - Mão livre	10.325	10.325	M5 - Mão restrita	10.935	10.935
5		10.325	10.325		10.935	10.935
4		-	-		-	-
3		-	-		-	-

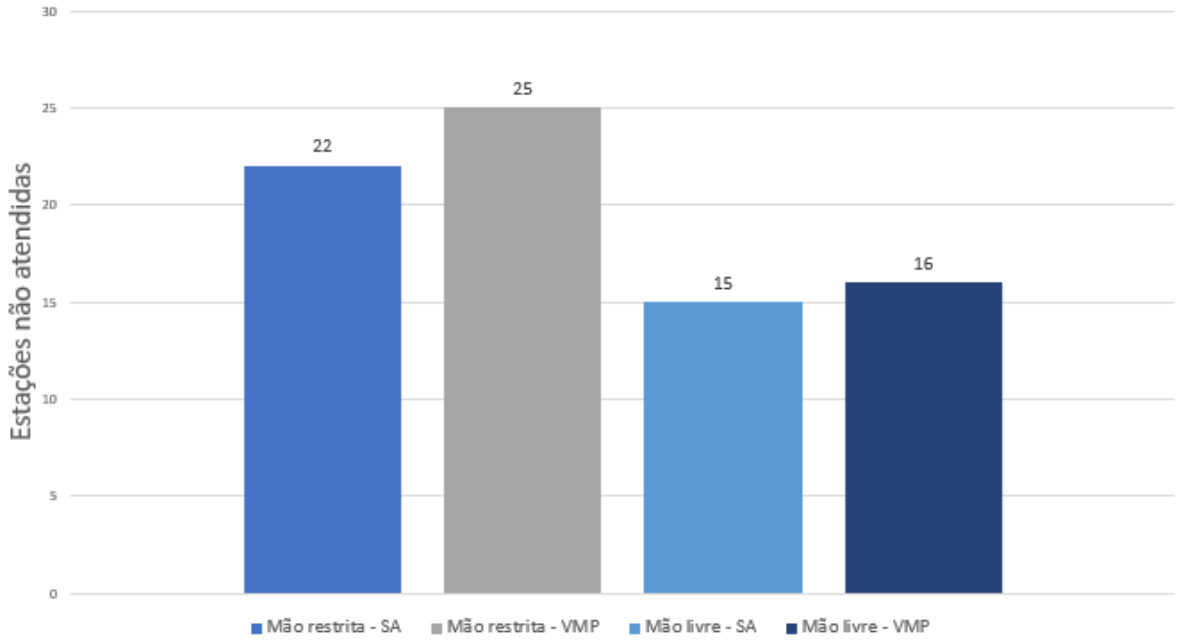
Fonte: Elaboração própria (2023).

Com a análise da tabela comparativa, pode-se inferir que o método do *Simulated Annealing* encontrou os resultados para as distâncias percorridas, uma vez que esses valores foram inferiores aos apresentados pelo método do Vizinho Mais Próximo. Assim, calcula-se que o erro percentual apresentado pelo algoritmo com o método do Vizinho Mais próximo, em comparação com o *Simulated Annealing*, foi igual a 3,85% no cenário sem restrições no *line supply* e 5,58% no cenário com restrições.

Desse modo, revela-se que o espaço limitado de busca do método do VMP tende a retornar soluções sub-ótimas em cenários com parâmetros excessivamente restrito (como no caso da matriz de janelas de tempo M3). Nesses casos, a met-heurística *Simulated Annealing* destaca-se como superior.

Outro ponto de referência comparativa no que tange à qualidade da solução é ao atendimento da demanda. Ao analisar a Figura 43, pode-se verificar que o método do Vizinho Mais Próximo apresentou um número maior de estações não atendidas durante o processo de reabastecimento da linha de montagem em comparação com o método *Simulated Annealing* (41 para o primeiro e 37 para o segundo).

Figura 43 – Comparação de estações não atendidas



Fonte: Elaboração própria (2023).

A diferença nesse desempenho, especialmente nos cenários com restrições no *line supply*, pode ser atribuída à abordagem de busca utilizada por cada um. Uma vez que o método do Vizinho Mais Próximo tem um horizonte de busca limitado, esse pode ser inadequado para cenários complexos com diversas restrições. Em contraste, o *Simulated Annealing* emprega uma abordagem mais exploratória e é capaz de escapar de mínimos locais, permitindo uma busca mais abrangente.

Com isso, a meta-heurística *Simulated Annealing* se destaca ao minimizar o número de estações não atendidas, principalmente em ambientes com restrições no *line supply*. Isso verifica-se ao comparar a porcentagem de demandas não atendidas nos cenários restritos:

- Nos cenários com limitações no *line supply*, o método do *Simulated Annealing* apresenta 59,46% das estações não atendidas.
- Nos cenários com limitações no *line supply*, o método do Vizinho Mais Próximo apresenta 60,98% das estações não atendidas.

Em relação ao uso de recursos de sistema, o método baseado na meta-heurística *Simulated Annealing* tende a requerer uma quantidade de memória significativamente maior do que o Vizinho Mais Próximo. A métrica RSS, que representa a quantidade de memória física (RAM) que um processo está usando, geralmente é mais elevada no SA devido à complexidade de suas operações. O VMS, que mede a quantidade total de memória virtual alocada, também pode ser mais alta para o *Simulated Annealing*, em função da necessidade do algoritmo de explorar uma ampla

gama de soluções.

A comparação entre o uso das duas memórias pode ser verificada na Figura 44, demonstrando a maior alocação geral no solucionador com a *Simulated Annealing* aplicada.

Figura 44 – Comparação de usos de memórias

		Simulated Annealing	*Simulated Annealing*	Vizinho Mais Próximo	Vizinho Mais Próximo		*Simulated Annealing*	*Simulated Annealing*	Vizinho Mais Próximo	Vizinho Mais Próximo
Tow trains		*Memory RSS Used (KB)*	*Memory VMS Used (KB)*	*Memory RSS Used (KB)*	*Memory VMS Used (KB)*		*Memory RSS Used (KB)*	*Memory VMS Used (KB)*	*Memory RSS Used (KB)*	*Memory VMS Used (KB)*
3		-	-	-	-		-	-	-	-
4	M1 - Mão livre	90.028	301.620	90.148	301.664	M1 - Mão restrita	89.836	301.540	90.144	301.668
5		91.092	301.836	91.152	301.732		91.088	301.848	91.096	301.688
6		91.288	301.960	90.920	301.364		90.900	301.476	90.912	301.372
7		91.172	301.844	91.040	301.480		91.268	302.076	90.940	301.284
8		91.828	302.456	91.732	302.540		90.696	302.068	91.588	302.296
Tow trains		*Memory RSS Used (KB)*	*Memory VMS Used (KB)*	*Memory RSS Used (KB)*	*Memory VMS Used (KB)*		*Memory RSS Used (KB)*	*Memory VMS Used (KB)*	*Memory RSS Used (KB)*	*Memory VMS Used (KB)*
3		-	-	-	-		-	-	-	-
4	M2 - Mão livre	90.132	301.816	90.112	301.660	M2 - Mão restrita	89.984	301.624	89.996	301.692
5		90.776	301.280	90.676	301.172		90.964	301.832	90.860	301.264
6		90.916	301.468	90.960	301.448		91.116	301.940	91.184	301.572
7		90.916	301.348	91.380	302.116		91.540	302.232	91.076	301.580
8		91.448	301.848	91.260	301.740		90.712	302.064	91.448	302.172
Tow trains		*Memory RSS Used (KB)*	*Memory VMS Used (KB)*	*Memory RSS Used (KB)*	*Memory VMS Used (KB)*		*Memory RSS Used (KB)*	*Memory VMS Used (KB)*	*Memory RSS Used (KB)*	*Memory VMS Used (KB)*
3		-	-	-	-		-	-	-	-
4	M3 - Mão livre	89.872	301.536	90.100	301.784	M3 - Mão restrita	89.992	301.688	90.076	301.716
5		91.008	301.856	90.972	301.368		90.636	301.128	90.868	301.100
6		90.796	301.344	91.208	301.608		91.124	301.796	91.124	301.436
7		91.212	301.884	91.068	301.688		90.928	301.492	91.280	301.868
8		91.452	301.912	91.492	302.212		91.576	302.228	91.232	301.660
Tow trains		*Memory RSS Used (KB)*	*Memory VMS Used (KB)*	*Memory RSS Used (KB)*	*Memory VMS Used (KB)*		*Memory RSS Used (KB)*	*Memory VMS Used (KB)*	*Memory RSS Used (KB)*	*Memory VMS Used (KB)*
3		-	-	-	-		-	-	-	-
4	M4 - Mão livre	90.124	301.668	90.124	301.632	M4 - Mão restrita	90.096	301.772	89.884	301.508
5		91.172	302.016	91.088	301.524		90.548	301.148	90.764	301.312
6		91.336	302.008	90.736	301.176		91.236	302.048	90.936	301.360
7		91.312	302.084	91.200	301.892		91.324	302.004	91.144	301.968
8		91.936	302.428	91.436	302.136		91.596	302.236	91.420	301.852
Tow trains		*Memory RSS Used (KB)*	*Memory VMS Used (KB)*	*Memory RSS Used (KB)*	*Memory VMS Used (KB)*		*Memory RSS Used (KB)*	*Memory VMS Used (KB)*	*Memory RSS Used (KB)*	*Memory VMS Used (KB)*
3		-	-	-	-		-	-	-	-
4	M5 - Mão livre	89.908	301.572	90.168	301.672	M5 - Mão restrita	90.052	301.588	90.132	301.656
5		91.220	301.952	90.820	301.396		91.048	301.776	91.116	301.608
6		91.208	302.016	90.868	301.312		91.164	301.848	90.964	301.304
7		91.616	302.312	91.224	301.936		91.232	302.016	91.204	302.052
8		92.064	302.696	91.504	302.212		90.596	301.996	91.448	301.904

Fonte: Elaboração própria (2023).

Ainda que o *Simulated Annealing* tenha apresentado os maiores valores de memórias alocadas, o Vizinho Mais Próximo apresentou-se superior em alguns cenários. Isso ocorre, pois em alguns cenários com restrições complexas, a execução do Vizinho Mais Próximo torna-se mais complicada, levando a um aumento temporário nos requisitos de memória à medida que o algoritmo tenta lidar com essas restrições. Ainda assim, as ocorrências com memórias alocadas superiores no Vizinho Mais Próximo correspondem apenas a 32% das ocorrências, indicando o caráter mais custoso do *Simulated Annealing* nos cenários gerais.

Em paralelo, o número de *tow trains* utilizados é uma métrica importante para a eficiência da operação. O método do Vizinho Mais Próximo tende a gerar soluções que exigem um número menor de veículos, no entanto, pode levar a soluções sub-ótimas de distâncias percorridas devido à falta de exploração abrangente do espaço de solução. O método *Simulated Annealing*, com uma abordagem mais aberta e exploratória, pode aumentar o número de *tow trains* utilizados para otimizar a eficiência geral.

A comparação entre ambos os métodos pode ser evidenciada na Figura 45, demonstrando que não há diferenças entre os números de *tow trains* obtidos como resultados. Assim, verifica-se que as duas soluções conseguiram obter valores ótimos para esse parâmetro, uma vez que esses apresentaram consistências entre ambos os métodos e os testes com frotas inferiores (4 ou 3 *tow trains* apresentaram falhas no atendimento da demanda, como apresentado anteriormente).

Figura 45 – Comparação de frota de *tow trains*

Simulated Annealing	**Vizinho Mais Próximo**		***Simulated Annealing***	**Vizinho Mais Próximo**
Número de veículos mínimos para atender a demanda		M1 - Mão restrita	Número de veículos mínimos para atender a demanda	
5	5		5	5
Número de veículos mínimos para atender a demanda		M2 - Mão restrita	Número de veículos mínimos para atender a demanda	
5	5		5	5
Número de veículos mínimos para atender a demanda		M3 - Mão restrita	Número de veículos mínimos para atender a demanda	
5	5		5	5
Número de veículos mínimos para atender a demanda		M4 - Mão restrita	Número de veículos mínimos para atender a demanda	
5	5		5	5
Número de veículos mínimos para atender a demanda		M5 - Mão restrita	Número de veículos mínimos para atender a demanda	
4	4		4	4

Fonte: Elaboração própria (2023).

Em última análise, a escolha entre os métodos *Simulated Annealing* e Vizinho Mais Próximo depende dos objetivos da otimização. O *Simulated Annealing* é uma escolha melhor quando a qualidade da solução é fundamental, mesmo que isso signifique custos computacionais mais altos em termos de tempo e recursos. Por outro lado, o Vizinho Mais Próximo é preferível quando a eficiência computacional é primordial, e a qualidade da solução pode ser comprometida em prol da velocidade.

Em virtude dos valores observados nas análises anteriores, reitera-se a validade do método *Simulated Annealing* como meta-heurística adotada para o algoritmo, uma vez que os custos computacionais não foram tão dispendiosos em comparação com o Vizinho Mais Próximo e as soluções foram superiores.

5 CONSIDERAÇÕES FINAIS

Este estudo dedicou-se ao desenvolvimento de um algoritmo voltado para o planejamento de rotas de *tow trains* em ambientes de linhas de montagem automotivas sob o sistema JIT. O objetivo primordial foi aprimorar a eficiência operacional, visando a redução das distâncias totais percorridas pelos *tow trains* e a minimização do número de veículos necessários para cumprir as entregas planejadas durante o reabastecimento da linha de montagem.

Para cumprir tais objetivos, ao longo desse trabalho, uma revisão abrangente da literatura foi realizada, abordando métodos de resolução de problemas de roteirização comumente empregados em contextos de reabastecimento de linhas de montagem. Em paralelo a isso, os conceitos apresentados na revisão bibliográfica desempenham um papel fundamental ao concentrar definições e temas que envolvem os problemas de roteirização, muitas vezes não facilmente acessíveis. Esta abordagem teórica destacou-se por enriquecer o conteúdo da pesquisa ao fornecer uma base sólida e detalhada para a construção do algoritmo e análise dos resultados.

A construção deste algoritmo baseou-se, em primeiro plano, na meta-heurística *Simulated Annealing*, reconhecida por sua capacidade de explorar soluções em espaços extensos de busca. Essa característica mostrou-se relevante no cenário de otimização global aqui proposto, uma vez que permitiu a obtenção satisfatória dos ótimos globais nos cenários testados.

A implementação desse método de busca deu-se por meio de uma abordagem integrada de programação em *Python*, aliada à biblioteca *OR-Tools* e essa estratégia ofereceu a flexibilidade necessária para lidar com os cenários complexos aqui definidos, marcados por várias condicionais e restrições. Além disso, destaca-se que a aplicação prática de meta-heurísticas em *Python*, em paralelo com uma biblioteca especializada como o *OR-Tools*, é uma estratégia que ainda não é amplamente adotada na solução de problemas de roteirização e, ao validar essa abordagem de integração, o presente trabalho contribui para o crescente reconhecimento de que a combinação das duas ferramentas é uma opção altamente eficaz para solucionar problemas de roteirização.

Um dos diferenciais deste trabalho em relação a problemas comuns de roteirização é a inclusão de múltiplas restrições e condicionais em um mesmo ambiente de estudo. Dentre essas, destacam-se as janelas de tempo rigorosas, que refletem a necessidade de cumprir prazos rígidos de entrega, o reabastecimento em supermercados específicos para garantir o fornecimento constante de peças, as restrições de direção das ruas no entorno da área de *line supply*, as quais impactam diretamente nas possibilidades de trajeto, bem como as condições variáveis de

demanda e a capacidade limitada dos *tow trains*. Essa abordagem inovadora visa simular situações reais de alta complexidade encontradas na indústria automotiva e, ao longo do desenvolvimento da pesquisa, validou-se a possibilidade de reunir todas as restrições em um único algoritmo solucionador que obteve respostas satisfatórias para os ambientes simulados.

Ainda, para avaliar a eficácia e o custo computacional do algoritmo, não apenas a meta-heurística *Simulated Annealing* foi utilizada, mas também a heurística do Vizinho Mais Próximo. Essa análise comparativa permitiu uma avaliação detalhada das vantagens e desvantagens de ambas as abordagens, além de fornecer dados sobre o custo computacional associado a cada uma delas. A escolha da *Simulated Annealing* como técnica principal foi justificada pela sua capacidade de explorar soluções em regiões de amplo espaço de busca, obtendo ótimos globais em todos os cenários simulados. Além disso, comprovou-se que, ainda que essa meta-heurística possua um custo computacional agregado relativamente mais alto do que o Vizinho Mais Próximo, esse não inviabilizada a escolha da *Simulated Annealing*, uma vez que essa apresentou resultados satisfatórios.

Ao realizar o estudo, foram simulados 120 cenários distintos no algoritmo, proporcionando uma análise abrangente para testar a validade e o comportamento do código em diversas condições de *input*, sendo gerados a partir de quatro estratégias variadas: restrições no *line supply*, aumentar e reduzir janelas de tempo e alterar a frota de *tow trains*.

Essa extensa análise comparativa contribuiu para verificar a eficácia do algoritmo em diferentes contextos e cenários aplicáveis a uma montadora de automóveis. Com isso, tal abordagem de simulações proporcionou uma compreensão mais aprofundada do comportamento do algoritmo sob diversas condições, fornecendo uma base sólida para a interpretação e generalização dos resultados obtidos.

Quanto a tais resultados, percebe-se que esses revelam que o algoritmo desenvolvido é capaz de otimizar consideravelmente as operações de reabastecimento em linhas de montagem automotivas. A redução das distâncias totais percorridas e a otimização do número de *tow trains* empregados demonstram a eficácia da abordagem, ao passo que cumpre-se o objetivo inicial do estudo.

Em específico, no que diz respeito aos resultados relacionados às alterações nas janelas de tempo, observa-se um *trade off* crucial a ser considerado para a programação dos processos de roteirização em linhas de montagem automotivas. Inicialmente, a redução das janelas de tempo mostra-se como uma estratégia eficaz para otimizar o tempo total alocado ao reabastecimento da linha de montagem, contudo, a imposição de janelas de tempo excessivamente restritas pode acarretar desafios significativos no que se refere ao cumprimento pontual das demandas das estações de trabalho.

Esse *trade off* evidencia a necessidade de encontrar um equilíbrio adequado ao ajustar as janelas de tempo, considerando não apenas a otimização do tempo total, mas também a manutenção da eficiência operacional e da produtividade.

De maneira semelhante, um *trade off* apresentou-se no que tange à redução do número de veículos na frota de *tow trains*, uma vez que esta, inicialmente se apresenta como uma estratégia para otimizar recursos e potencialmente reduzir custos operacionais, contudo, assim se realizada de maneira excessiva pode resultar em falhas no atendimento da demanda. Portanto, ao buscar otimizar a frota de veículos, é crucial encontrar um equilíbrio que permita a redução de custos sem comprometer a capacidade do sistema de atender de maneira eficiente às demandas da linha de montagem.

Considerando os *trade offs* discutidos em relação às janelas de tempo, frota de veículos e método de busca, a análise abrangente dos resultados revela que o cenário mais eficiente e equilibrado, atendendo aos objetivos fundamentais do algoritmo, é representado pela matriz de janelas de tempo M4 sem restrições no *line supply*. Esta matriz, caracterizada por intervalos médios em comparação com os demais cenários, demonstrou ser uma escolha balanceada, evitando os extremos de restrição excessiva ou falta de restrição. No que diz respeito à frota de veículos, a configuração mais eficaz foi alcançada com cinco *tow trains*, escolha que equilibra a redução de custos operacionais com a manutenção de uma capacidade adequada para atender às demandas das estações de trabalho dentro das janelas de tempo estabelecidas.

Com isso, evidencia-se que os *trade offs* exercem um papel crucial no gerenciamento da frota de *tow trains*, nos custos de produção e na logística de montadoras de automóveis. Ao reduzir o número de *tow trains*, busca-se economizar operacionalmente, contudo, esse declínio deve ser ponderado para assegurar a capacidade de atender à demanda. A gestão eficaz das janelas de tempo, embora otimize o tempo total, pode gerar pressões no cumprimento de prazos, aumentando custos devido a paradas na produção ou penalidades contratuais. Além disso, esses *trade offs* impactam diretamente a eficiência operacional, influenciando a produtividade e a logística de movimentação de peças, uma vez que a flexibilidade do sistema é essencial para lidar com variações na demanda.

Pode-se destacar, também, que uma grande vantagem do algoritmo aqui desenvolvido é que, devido a sua flexibilidade, esse não se limita apenas à sua aplicação em linhas de montagem automotivas. A análise de roteirização, então, pode ser estendida para diversos outros contextos logísticos e de transporte, onde a necessidade de otimizar rotas e reduzir custos é essencial. A modularidade e adaptabilidade das abordagens de programação utilizadas abrem oportunidades para futuras expansões e personalizações do algoritmo, tornando-o uma ferramenta versátil para enfrentar uma ampla gama de desafios de roteirização.

Por fim, mais um dos pontos notáveis deste estudo é o equilíbrio entre a complexidade do problema abordado e a eficiência do algoritmo desenvolvido. Embora a roteirização de *tow trains* envolva um grande número de variáveis e restrições, o algoritmo conseguiu lidar com essa complexidade de forma eficaz, fornecendo soluções de alta qualidade em um tempo razoável.

No entanto, é importante reconhecer que ainda há espaço para aprimoramentos. Com a continuidade desse trabalho em pesquisas futuras, recomenda-se a exploração de diferentes meta-heurísticas no algoritmo, a fim de avaliar a adequação do desse em variados contextos e buscar soluções ainda mais eficientes. A comparação de diferentes abordagens de otimização pode revelar as vantagens, desvantagens e as aplicações ideais de cada técnica nos cenários alvos da análise.

Além disso, recomenda-se a aplicação de dados reais de uma linha de montagem automotiva, a qual permitirá uma validação mais precisa das otimizações propostas e a identificação de desafios específicos associados a um ambiente de produção controlado. A colaboração com empresas do setor automotivo pode abrir portas para a implementação prática do algoritmo, proporcionando um ambiente de teste para seu desempenho em cenários reais e para a verificação da necessidade de eventuais ajustes.

Por fim, destaca-se que este trabalho oferece uma contribuição substancial para o entendimento de problemas de roteirização de *tow trains* em linhas de montagem automotivas e um algoritmo eficiente para obter a solução ótima em cenários complexos de análise. Sua abordagem inovadora, análise extensa e uso eficiente de recursos computacionais proporcionam uma base sólida para futuras pesquisas e aplicações práticas no mundo real, possuindo potencial para impactar positivamente a eficiência e a sustentabilidade das operações logísticas em várias indústrias.

REFERÊNCIAS

ALDEANO, J. P. C. C. T. **A decision support system in shuttle services managing**. 2020. Dissertação (Mestrado em Engenharia Informática) — Escola de Ciências e Tecnologia, Universidade de Évora, Évora, 2020.

ALI, M. **Using dynamic programming and unsupervised learning to optimize material flow in assembly line supermarket**: a case study of volvo powertrain at skövde. 7 p. Doctoral thesis in Information Technology — Skövde: University of Skövde, 2019.

ARAUJO, H. A. **Algoritmo Simulated Annealing**: uma nova abordagem. 2009. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal de Santa Catarina, Florianópolis, 2001.

ARENALES, M. e. a. **Pesquisa Operacional**. Rio de Janeiro: Elsevier, 2015.

ARI, S. T.; HOANG, K. Shared computer-integrated manufacturing for various types of production environment. **International Journal of Operations Production Management**, v. 15, n. 5, p. 95–108, 1995.

BATINNI, D.; BOYSEN, N.; EMDE, S. Just-in-time supermarkets for part supply in the automobile industry. **Journal of Management Control**, v. 24, n. 2, p. 209–217, 2013.

BEBER, J. **Um método para a implementação de um sistema enxuto de abastecimento ship to line**: um estudo de caso. 2009. Dissertação (Mestrado em Engenharia Mecânica) — Universidade Federal de Santa Catarina, Florianópolis, 2009.

BELLMORE, M.; NEMHAUSER, G. L. The traveling salesman problem: a survey. **JSTOR**, v. 16, n. 3, p. 538–558, 1968.

BROGIATO, G. C. **Programação linear, linear inteira e os algoritmos Simplex e Branch and Bound**: Problemas e aplicações em otimização. Trabalho de Conclusão de Curso (Graduação em Engenharia da Computação) — Faculdade de Engenharia, Universidade Federal de São Carlos, São Carlos, 2021.

CABRAL, G. T. **Um otimizador estático para a linguagem Python**. 2013. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) — Faculdade de engenharia, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2013.

CARNEIRO, F. M. **Avaliação de métodos heurísticos para a solução do problema de programação flow shop com tempos de setup assimétricos e dependentes da sequencia**. 2010. Dissertação (Mestrado em Engenharia de Produção) — Escola de Engenharia, Universidade de São Paulo, São Carlos, 2010.

CAUX, C.; DURIEUX, S.; SAAIDIA, M. A survey on supermarket concept for just-in-time part supply of mixed model assembly lines. *In*: **Proceedings** of the 10th INTERNATIONAL CONFERENCE OF MODELING AND SIMULATION. Nancy, France, 2014. Disponível em: https://hal.science/hal-01166624/document. Acesso em: 28 mar. 2023.

CORDEAU, J. et al. **Vehicle routing**. Amsterdã: Elsevier, 2007.

CUNHA, C. B.; WU, L. O problema da roteirização periódica de veículos. **TRANSPORTES**, v. 16, n. 1, p. 5–16, 2008.

DANTZIG, G. B.; RAMSER, J. H. The truck dispatching problem. **Management Science**, v. 6, n. 1, p. 90–91, 1959.

DEGAN, R. J. Fordism and taylorism are responsible for the early success and recent decline of the u.s. motor vehicle industry. **Globo Advantage**, v. 1, n. 81, p. 01–30, 2011.

DIAZ, A. A. A. **Optimization models and solution methods for inventory routing problems**. 2020. Dissertação (Doutorado em Engenharia de Produção) — Centro de Ciências Exatas e Tecnologia, Universidade Federal de São Carlos, São Carlos, 2020.

EMDE, S.; BOYSEN, N. Optimally routing and scheduling tow trains for jit-supply of mixed-model assembly lines. **European Journal of Operational Research**, v. 217, n. 2, p. 287–299, 2011.

EMDE, S.; DIEFENBACH, H.; GLOCK, C. H. Loading tow trains ergonomically for just-in-time part supply. **European Journal of Operational Research**, v. 284, n. 1, p. 325–344, 2020.

EMDE, S.; GENDREAU, M. Scheduling in-house transport vehicles to feed parts to automotive assembly lines. **European Journal of Operational Research**, v. 260, n. 1, p. 255–267, 2017.

EMDE, S.; POLTEN, L. Sequencing assembly lines to facilitate synchronized just-in-time part supply. **Journal of Scheduling**, v. 22, p. 607–621, 2019.

FACCIO, M. et al. Design and simulation of assembly line feeding systems in the automotive sector using supermarket, kanbans and tow trains: A general framework. **Journal of Management Control**, v. 24, n. 2, p. 187–208, 2013.

FERNANDES, A. S. **Optimization algorithms for the inventory routing problem**. 2016. Dissertação (Mestrado em Engenharia Eletrotécnica e de Computadores) — Faculdade de Engenharia, Universidade do Porto, Porto, 2016.

FERNANDES, F. R. S. **Uma nova abordagem para o problema da patrulha escolar**: formulação matemática e metaheurísticas. 2019. Dissertação (Mestrado em Ciência da Computação) — Universidade do Estado do Rio Grande do Norte e Universidade Federal Rural do Semi-Árido, 2019.

FUCHIGAMI, H. Y. **Métodos heurísticos construtivos para o problema de programação da produção em sistemas flow shop híbridos com tempos de preparação das máquinas assimétricos e dependentes da seqüência**. 2005. Dissertação (Mestrado em Engenharia de Produção) — Escola de Engenharia, Universidade de São Paulo, São Carlos, 2005.

GOLDBARG, M. C.; LUN, H. P. L. **Otimização Combinatória e Programação Linear**: modelos e algoritmos. Rio de Janeiro: Elsevier, 2005.

GOLZ, J. et al. Part feeding at high-variant mixed-model assembly lines. **Flexible Services and Manufacturing Journal**, v. 24, n. 2, p. 119–141, 2012.

GOMES, H. A. S. **Utilização da metaheurística Simulated Annealing no problema de alocação de pessoal em empresas de transporte coletivo por ônibus**. 2003. Dissertação (Mestrado em Engenharia de Transportes) — Centro de Tecnologia, Universidade Federal do Ceará, Fortaleza, 2003.

HASLE, G.; KLOSTER, O. **Industrial vehicle routing problems.** Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. 397-435 p.

JOURDAN, L.; BASSEUR, M.; TALBI, E. G. Hybridizing exact methods and metaheuristics: A taxonomy. **European Journal of Operational Research**, v. 199, n. 3, p. 620–629, 2009.

KIRKPATRICK, C.; GELATT, C. D.; VECCHI, M. Optimization by simulated annealing. **Science**, v. 220, n. 4598, p. 671–680, 1983.

KRUK, S. **Pratical Python ai projects**. Rochester, Michigan: Apress, 2018.

LAMB, T. **Ship design and construction**. New Jersey: The Society of Naval Architects and Marine Engineers, 2003.

LEMOS, L. A. Fordismo, toyotismo e novos paradigmas econômicos. **Análise**, v. 1, n. 5, p. 513–521, 1991.

MACHADO, R. P.; FRANCO, M. I.; BERTAGNOLLI, S. C. **Desenvolvimento de software III**: Programação de sistemas web orientada a objetos em Java. Porto Alegre: Bookman, 2016.

MALDANER, L. **Desenvolvimento de software acadêmico para engenharia química utilizando linguagem Python**: separador de mistura binária. 2019. Trabalho de Conclusão de Curso (Graduação em Engenharia Química) — Centro de Tecnologia, Universidade Tecnológica Federal do Paraná, Francisco Beltrão, 2019.

MULLER, G. Local vs. global optimization in syntax: a case study. **IDS Mannheim**, v. 2, p. 82–91, 2004.

OLHAGER, J. The role of decoupling points in value chain management. *In*: **Proceedings** CONTRIBUTIONS TO MANAGEMENT SCIENCE. Linköping, Sweden, 2011. Disponível em: https://link.springer.com/chapter/10.1007/978-3-7908-2747-7_2# citeas. Acesso em: 21 mar. 2023.

OLIPHANT, T. E. Python for scientific computing. **Computing in Science & Engineering**, v. 9, p. 10–20, 2007.

PERRON, L.; FURNON, V. **OR-Tools**. 2023. Disponível em: https://developers.google.com/optimization/.

RIBEIRO, A. L. R. **Simulated Annealing Aplicado ao problema do Caixeiro Viajante**. Trabalho de Conclusão de Curso (Graduação em Engenharia de Produção) — Faculdade de Engenharia, Universidade Federal de Juiz de Fora, Juiz de Fora, 2018.

ROSSI, A. C. **Análise dos processos de otimização no projeto de um navio Roll-on/Roll-off**. 2018. Trabalho de Conclusão de Curso (Graduação em Engenharia Naval), Centro Tecnológico de Joinville — Universidade Federal de Santa Catarina, Joinville, 2018.

SANTOS, I. C. V. **Utilização da metaheurística simulated annealing para a otimização da programação de turnos dos funcionários de uma loja varejista**. 2016. Trabalho de Conclusão de Curso (Graduação em Engenharia de Produção), Centro de engenharia — Universidade Federal de Ouro Preto, Ouro Preto, 2016.

SOLOMON, M. M. Algorithms for the vehicle routing and scheduling problems with time window constraints. **Operations Research**, v. 15, n. 2, p. 254–265, 1987.

SONG, J. S.; ZIPKIN, P. **Supply chain operations**: assemble-to-order systems. handbooks in operations research and management science. Amsterdã: Elsevier, 2003.

SULE, D. R. **Manufacturing facilities**: location, planning, and design. 3. ed. Boca Raton: CRC Press, 2009.

TAYLOR, F. **Princípios de administração científica**. 8. ed. São Paulo: ATLAS S.A, 1990.

VERCELLINO, C. **A two-step optimization approach for a stochastic multi stage capacitated vehicle routing problem**. 2019. Dissertação (Mestrado em Engenharia Matemática) — Departamento de Ciências Matemáticas, Politecnico di Torino, Itália, 2020.