UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

KEMUEL MARVILA

APLICAÇÃO DE TÉCNICAS DE OTIMIZAÇÃO PARA O PROBLEMA DE SELEÇÃO DE PEDIDOS ÓTIMA

Ouro Preto
2026

KEMUEL MARVILA

APLICAÇÃO DE TÉCNICAS DE OTIMIZAÇÃO PARA O PROBLEMA DE SELEÇÃO DE PEDIDOS ÓTIMA

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Pablo Luiz Araújo Munhoz

Ouro Preto
2026

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

M391a Marvila, Kemuel.
Aplicação de técnicas de otimização para o problema de seleção de pedidos ótima. [manuscrito] / Kemuel Marvila. - 2026.
96 f.: il.: color., tab.. + Pseudocódigos.

Orientador: Prof. Dr. Pablo Luiz Munhoz.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Biológicas. Graduação em Ciência da Computação .

1. Otimização combinatória. 2. Metaheurísticas. 3. Programação. I. Munhoz, Pablo Luiz. II. Universidade Federal de Ouro Preto. III. Título.

CDU 681.5

Bibliotecário(a) Responsável: Angela Maria Raimundo - SIAPE: 1.644.803

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E BIOLOGICAS
DEPARTAMENTO DE COMPUTACAO

FOLHA DE APROVAÇÃO

Kemuel Marvila

APLICAÇÃO DE TÉCNICAS DE OTIMIZAÇÃO PARA O PROBLEMA DE SELEÇÃO DE PEDIDOS ÓTIMA

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Ciência da Computação

Aprovada em 28 de Julho de 2026.

Membros da banca

Pablo Luiz Araújo Munhoz (Orientador) - Doutor - Universidade Federal de Ouro Preto
Puca Huachi Vaz Penna (Examinador) - Doutor - Universidade Federal de Ouro Preto
Pedro Henrique Gonzalez Silva (Examinador) - Doutor - Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa em Engenharia - COPPE/UFRJ

Pablo Luiz Araújo Munhoz, Orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 28/07/2026.

Documento assinado eletronicamente por **Pablo Luiz Araujo Munhoz**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 29/07/2026, às 21:13, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **1147685** e o código CRC **29EEDF58**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: 3135591692 - www.ufop.br

Agradecimentos

Parecia um loop infinito, mas o *break;* finalmente chegou. Que jornada foi esta graduação! Além da Ciência da Computação, ela me proporcionou amizades, experiências e aprendizados que levarei para toda a vida.

Agradeço primeiramente a Deus e à comunidade da Igreja Adventista de Ouro Preto, pelo acolhimento e pela força espiritual.

Esta conquista não seria possível sem minha base: minha família. Aos meus pais, Nicacio e Edna Lucia, e à minha irmã, Dominik, obrigado por serem o *sudo* da minha vida, me dando permissão e suporte incondicional para cada passo. Sem vocês, nada disso seria real.

À UFOP e ao DECOM, expresso minha gratidão pelo ensino de qualidade. Um agradecimento especial ao meu orientador, Prof. Dr. Pablo Luiz Araújo Munhoz, por toda a paciência, dedicação e orientação fundamentais para fazer este TCC enfim compilar sem erros.

Aos amigos de código, em especial ao Tiago Gomes, agradeço pela parceria inabalável nos projetos e por aguentar dois anos de tutoria.

Minha eterna gratidão à gloriosa República Exílio. Se Ouro Preto foi a faculdade, a Exílio foi o lar. Obrigado aos meus irmãos de república, ex-alunos, homenageados e moradores, pelas boas conversas que ajudaram a "debugar" a vida em momentos difíceis, e por cada momento excelente e incompreensível. Vocês são meu *git commit -m 'melhores anos da minha vida'*. Como o hino já diz, "quanto mais dessa eu bebo, mais dessa eu quero virar", e essa república eu quero levar pra sempre comigo.

E, claro, um salve às repúblicas parceiras que fizeram o caos ser muito mais divertido: Imprevisto, Melindrosa, Quinta Negra, Teoria do Caos, Rosa Xoque, Mistura Perfeita, M&Ms, IML e Troia. Obrigado por cada festa e cada porta aberta.

Encerro este capítulo com o coração cheio de gratidão. Que venha o próximo "Hello, World!".

Resumo

Este trabalho aborda o Problema da Seleção de Pedidos Ótima (PSPO), um problema intralogístico de otimização da coleta de pedidos (*order picking*) e formação de lotes (*order batching*) em centros de distribuição de comércio eletrônico (*e-commerce*). Fundamentada no desafio do LVII Simpósio Brasileiro de Pesquisa Operacional (SBPO) em parceria com o Mercado Livre, a problemática modela-se como um problema de Programação Fracionária *NP-Hard*, cuja função objetivo visa a maximizar a produtividade da onda de coleta (*wave*), expressa pela razão entre itens selecionados e corredores ativados, sob restrições de capacidade e estoque. Para sua resolução, desenvolveu-se o *OOSP-MetaSolver Framework* na linguagem Rust, integrando avaliação incremental de saldos em complexidade temporal $O(k)$, heurísticas construtivas orientadas a pedidos (*Order-Centric*) e a corredores (*Aisle-Centric*), exploração multi-vizinhança com o *Variable Neighborhood Descent* (VND) e as metaheurísticas GRASP Reativo (R-GRASP) e Otimização por Enxame de Partículas Binário Adaptativo Reativo (AR-BPSO) com paralelismo nativo. Conduziu-se avaliação experimental nas 50 instâncias oficiais do problema sob o limite fixo de 600 segundos. Os experimentos indicam que a razão de densidade pedidos/corredores ($R = \frac{\text{Ped.}}{\text{Cor.}}$) orienta a eficácia construtiva: quando $R \geq 1$, a consolidação por corredores apresenta desvio médio de 7,64% em menos de 0,05 segundos, margem reduzida para 1,71% pela busca local VND. Na avaliação global, o R-GRASP obteve desvio médio de 1,00% e convergiu para a melhor solução conhecida (*Best Known Solution*, BKS) em 46,00% das instâncias em 77,34 segundos. O AR-BPSO alcançou o menor desvio do estudo (0,79%) e convergiu para a BKS em 50,00% das instâncias em 247,57 segundos médios. No confronto com a literatura, as abordagens propostas apresentaram resultados superiores à heurística de referência do estado da arte em 9 instâncias (convergindo para a BKS em 5 delas) e encontraram soluções viáveis com desvios residuais inferiores aos de modelos exatos de Programação Linear Inteira onde estes não fecharam o *gap* dual. De modo geral, os resultados indicam a aplicabilidade do método para operações intralogísticas.

Palavras-chave: Otimização combinatória. *Order batching*. Programação fracionária. Metaheurísticas. Rust.

Abstract

This work addresses the Optimal Order Selection Problem (OOSP), an intralogistics optimization problem involving order picking and order batching in *e-commerce* fulfillment centers. Based on the real-world challenge of the 57th Brazilian Symposium on Operations Research (SBPO) in partnership with Mercado Livre, the problem is modeled as an *NP-Hard* Fractional Programming problem whose objective function seeks to maximize the operational productivity of a picking wave, expressed as the ratio between selected items and activated aisles, under capacity and inventory constraints. To solve it, the OOSP-MetaSolver Framework was developed in the Rust programming language, integrating incremental stock evaluation in $O(k)$ time complexity, adaptive constructive heuristics centered on orders (*Order-Centric*) and aisles (*Aisle-Centric*), multi-neighborhood exploration via Variable Neighborhood Descent (VND), and high-order meta-heuristics comprising Reactive GRASP (R-GRASP) and Adaptive Reactive Binary Particle Swarm Optimization (AR-BPSO) with native data parallelism. An experimental evaluation was conducted on the 50 official instances under a fixed execution time limit of 600 seconds. Experimental analysis indicates that the order-to-aisle density ratio ($R = \frac{\text{Ord.}}{\text{Ais.}}$) guides constructive efficiency: when $R \geq 1$, aisle-centric consolidation yields an average deviation of 7.64% in under 0.05 seconds, which is further reduced to 1.71% by the VND local search. In global evaluation, R-GRASP achieved an average deviation of 1.00% and converged to the Best Known Solution (BKS) in 46.00% of the instances in an average of 77.34 seconds. AR-BPSO achieved the lowest average deviation of the study (0.79%) and converged to the BKS in 50.00% of the instances in an average of 247.57 seconds. In comparison with scientific literature, the proposed approaches achieved superior results compared to state-of-the-art reference heuristics on 9 instances (closing the gap to the BKS in 5 of them) and found feasible solutions with lower residual gaps than exact Integer Linear Programming models where exact solvers failed to close the dual gap within the time limit. Overall, the results indicate the applicability of the method for intralogistics operations.

Keywords: Combinatorial optimization. *Order batching*. Fractional programming. Meta-heuristics. Rust.

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Abreviaturas e Siglas

AAC	*Automatic Algorithm Configuration*
ABNT	Associação Brasileira de Normas Técnicas
AC	*Aisle-Centric* (Construção Orientada a Corredores)
AC-A	*Aisle-Centric Adaptive* (Construção Orientada a Corredores Adaptativa)
AC-R	*Aisle-Centric Randomized* (Construção Orientada a Corredores Aleatorizada)
ACO	*Ant Colony Optimization*
ALNS	*Adaptive Large Neighborhood Search*
AR-BPSO	*Adaptive Reactive Binary Particle Swarm Optimization*
BKS	*Best Known Solution*
BPB	*Branch-and-Price*
BPSO	*Binary Particle Swarm Optimization*
BRKGA	*Biased Random-Key Genetic Algorithm*
CPLEX	*C-Programming Linear Simplex*
DECOM	Departamento de Computação
DIR	Decodificador de Instâncias Reduzidas
EDD	*Earliest Due Date*
EMA	*Exponential Moving Average* (Média Móvel Exponencial)
GA	*Genetic Algorithms*
GB	*Gigabyte*
GIR	Gerador de Instâncias Reduzidas
GRASP	*Greedy Randomized Adaptive Search Procedure*
GS	*Generate-and-Solve*
ILS	*Iterated Local Search*
JOBASRP	*Joint Order Batching, Assignment, Sequencing and Routing Problem*

KDE	*K Desktop Environment*
LAHC	*Late Acceptance Hill Climbing*
LB	*Lower Bound* (Limite Inferior)
LSA3	*Location Selection Algorithm 3* (*Minimum Aisle Coverage*)
LSA7	*Location Selection Algorithm 7*
MILFP	*Mixed-Integer Linear Fractional Programming*
MILP	*Mixed-Integer Linear Programming*
OBP	*Order Batching Problem*
OC	*Order-Centric* (Construção Orientada a Pedidos)
OC-A	*Order-Centric Adaptive* (Construção Orientada a Pedidos Adaptativa)
OC-R	*Order-Centric Randomized* (Construção Orientada a Pedidos Aleatorizada)
OOSP	*Optimal Order Selection Problem*
OOSP-MetaSolver	*Optimal Order Selection Problem Metaheuristic Solver*
PLI	Programação Linear Inteira
PLIM	Programação Linear Inteira Mista
PNLI	Programação Não Linear Inteira
PSO	*Particle Swarm Optimization*
PSPO	Problema da Seleção de Pedidos Ótima
RAM	*Random Access Memory*
RCL	*Restricted Candidate List* (Lista Restrita de Candidatos)
R-GRASP	*Reactive Greedy Randomized Adaptive Search Procedure*
SA	*Simulated Annealing*
SBPO	Simpósio Brasileiro de Pesquisa Operacional
SKU	*Stock Keeping Unit*
SPO	Seleção de Pedidos Ótima
TS	*Tabu Search*

TSB	*Tabu Search Batching*
TSP	*Travelling Salesman Problem*
UB	*Upper Bound* (Limite Superior)
UFOP	Universidade Federal de Ouro Preto
VND	*Variable Neighborhood Descent*
WMS	*Warehouse Management System*

Lista de Símbolos

O	Conjunto de pedidos disponíveis ($o \in \{0, \ldots, \|O\| - 1\}$)
I	Conjunto de itens distintos ($i = 0, \ldots, \|I\| - 1$)
A	Conjunto de corredores do armazém ($a = 0, \ldots, \|A\| - 1$)
S	Espaço de busca de soluções viáveis do problema
T	Lista Tabu utilizada para registro de movimentos proibidos na Busca Tabu
s	Solução corrente ($s = (x_o, y_a)$) durante o processo de busca
s'	Solução vizinha ou candidata ($s' \in N(s)$)
s^*	Melhor solução viável encontrada durante a execução algorítmica
$N(s)$	Vizinhança da solução s no espaço de soluções
N_k	k-ésima estrutura de vizinhança no *Variable Neighborhood Descent* (VND)
d_{oi}	Demanda do item i pelo pedido o
q_{ai}	Disponibilidade de estoque do item i no corredor a
LB	Limite inferior de itens permitidos em uma onda de coleta (*wave*)
UB	Limite superior de itens permitidos em uma onda de coleta (*wave*)
x_o	Variável binária indicando se o pedido o foi selecionado (1) ou não (0)
y_a	Variável binária indicando se o corredor a foi visitado (1) ou não (0)
w_{oai}	Quantidade do item i do pedido o coletada no corredor a
Z	Função objetivo (produtividade operacional da onda de coleta)
$f(s)$	Valor da função objetivo associado à solução s
$\|O\|$	Cardinalidade do conjunto de pedidos (quantidade total de pedidos)
$\|I\|$	Cardinalidade do conjunto de itens (quantidade total de itens distintos)
$\|A\|$	Cardinalidade do conjunto de corredores (quantidade total de corredores)
R	Razão de densidade combinatória pedidos/corredores ($R = \frac{\text{Ped.}}{\text{Cor.}}$ ou $\frac{\|O\|}{\|A\|}$)

$I(p)$	Número total de itens demandados pelo pedido p
$A_{\min}(p)$	Limite inferior de corredores necessários para atender ao pedido p
$\Phi(pedido)$	Produtividade ou utilidade de um pedido na construção heurística
N_{itens}	Quantidade de itens únicos solicitados em um pedido
M_{itens}	Quantidade de tipos de itens diferentes armazenados em um corredor
$N_{pedidos}$	Quantidade total de pedidos selecionados em uma *wave*
$N_{corredores}$	Quantidade total de corredores visitados em uma *wave*
q_i	Quantidade demandada do item i em um pedido
q_i	Quantidade estocada do item i em um corredor
$D_0(i)$	Demanda global inicial do item i no *backlog*
$D(i)$	Demanda global pendente (ou residual) do item i durante a construção
$U_0(a)$	Utilidade estática inicial do corredor a baseada na demanda global
$U(a)$	Utilidade dinâmica do corredor a considerando apenas a demanda residual
α	Parâmetro de aleatorização para delimitação da Lista Restrita de Candidatos (RCL)
β	Taxa de suavização na Média Móvel Exponencial do GRASP Reativo (R-GRASP)
q	Expoente de amplificação da função heurística no R-GRASP
p_{OC}	Probabilidade de seleção da heurística construtiva orientada a pedidos (*Order-Centric*)
p_{AC}	Probabilidade de seleção da heurística construtiva orientada a corredores (*Aisle-Centric*)
τ	Parâmetro de expiração (*tenure*) na Lista Tabu
$iter_{\max}$	Limite máximo de iterações permitidas em heurísticas de busca local
$k_{\max}$	Tamanho máximo de amostra na exploração estocástica de vizinhanças
L	Tamanho da lista de memória de custos no *Late Acceptance Hill Climbing* (LAHC)

F_{hist}	Vetor circular de histórico de aptidões no *Late Acceptance Hill Climbing* (LAHC)
M	Tamanho da população (número de partículas no enxame AR-BPSO)
x_{ij}^t	Posição da i-ésima partícula na j-ésima dimensão na iteração t
v_{ij}^t	Velocidade da i-ésima partícula na j-ésima dimensão na iteração t
$v_{\max}$	Velocidade máxima permitida para as partículas no enxame ($[-v_{\max}, v_{\max}]$)
w	Fator de inércia da partícula no enxame (AR-BPSO)
c_1	Coeficiente de aceleração cognitiva (influência da memória individual da partícula)
c_2	Coeficiente de aceleração social (influência da melhor solução global do enxame)
r_1, r_2	Variáveis aleatórias uniformemente distribuídas no intervalo $[0, 1]$
$pbest_{ij}$	Melhor posição histórica individual alcançada pela i-ésima partícula na dimensão j
$gbest_j$	Melhor posição global histórica encontrada por todo o enxame na dimensão j
$O(k)$	Complexidade temporal proporcional ao número de elementos alterados k
$O(1)$	Complexidade temporal constante
$O(n)$	Complexidade temporal linear no tamanho da entrada n
$O(n^2)$	Complexidade temporal quadrática no tamanho da entrada n
$\mathbb{Z}_+$	Conjunto dos números inteiros não negativos
$\{0, 1\}$	Domínio de variáveis de decisão binárias
$\emptyset$	Conjunto vazio
$\in$	Pertence a um conjunto
$\forall$	Para todo (quantificador universal)
$\sum$	Operador de somatório
$\leq$	Relação de menor ou igual que
$\geq$	Relação de maior ou igual que

∞	Infinito
$\lfloor \ldots \rfloor$	Função parte inteira inferior (*floor*)

Sumário

1 Introdução

O comércio eletrônico (*e-commerce*) tem experimentado um crescimento exponencial nas últimas décadas, transformando os hábitos de consumo e impondo novos desafios à cadeia de suprimentos global. Neste cenário de alta competitividade, a eficiência logística tornou-se um fator determinante para a viabilidade operacional e competitividade do setor. Entre as etapas da cadeia logística, as operações intralogísticas, correspondentes àquelas executadas dentro dos armazéns e centros de distribuição, desempenham papel crítico na satisfação do cliente e na composição dos custos operacionais.

Dentre as atividades de armazém, a coleta de pedidos, ou *order picking*, é reconhecida na literatura como a etapa mais onerosa da operação, estimando-se que represente cerca de 60% dos custos operacionais totais de um centro de distribuição (Pardo *et al.*, 2024). O processo consiste em localizar e recuperar os itens solicitados pelos clientes nas prateleiras para posterior expedição. A ineficiência nesta etapa, ocasionada por deslocamentos excessivos e estratégias de roteamento inadequadas, afeta diretamente o tempo de ciclo do pedido (*lead time*) e a capacidade logística de processamento.

Para mitigar a ineficiência decorrente de percursos excessivos, adota-se a estratégia de agrupamento de pedidos em lotes, denominada na literatura como *Order Batching*. Essa abordagem insere-se entre os problemas de otimização combinatória operacionais na gestão de armazéns (Pardo *et al.*, 2024). A política consiste em consolidar múltiplos pedidos de clientes em lotes (*batches*) antes da coleta, permitindo que todos os itens do lote sejam retirados em uma única rota de visitação, o que otimiza o deslocamento do operador e reduz o tempo total de viagem por item (Pardo *et al.*, 2024).

Neste contexto, o presente trabalho aborda o Problema da Seleção de Pedidos Ótima (PSPO), com base nas instâncias e na modelagem matemática formulada no LVII Simpósio Brasileiro de Pesquisa Operacional (SBPO) em parceria com o Mercado Livre (Mercado Livre, 2025). A problemática distingue-se das formulações clássicas de minimização de distância ou tempo de viagem por caracterizar-se como um modelo de Programação Fracionária, cuja função objetivo visa a maximizar a produtividade da onda de coleta (*wave*), expressa pela razão entre o total de itens selecionados e o número de corredores ativados, sujeita a restrições operacionais de capacidade e disponibilidade de estoque.

Diante da complexidade combinatória *NP-Hard* do problema (Pardo *et al.*, 2024) e da não linearidade inerente à função objetivo fracionária, o estudo desenvolve e implementa a arquitetura algorítmica modular denominada *Optimal Order Selection Problem Metaheuristic Solver Framework* (*OOSP-MetaSolver Framework*) na linguagem Rust, estruturando heurísticas construtivas especializadas, exploração local multi-vizinhança e metaheurísticas reativas populacionais para a

sua resolução.

1.1 Justificativa

A gestão de armazéns constitui um dos pilares da operação do varejo digital moderno. Com a demanda por prazos de entrega reduzidos, a otimização da separação de pedidos torna-se crítica.

1.1.1 Motivação Prática

No contexto da logística e da gestão de cadeia de suprimentos, o processo de separação de produtos (*Order Picking*) constitui a etapa mais onerosa de um armazém, concentrando a maior parcela dos custos operacionais (Koster; Le-Duc; Roodbergen, 2007). A otimização dessa etapa figura como um fator determinante para a viabilidade e competitividade das empresas.

Para empresas de pequeno porte ou cenários com volume reduzido de pedidos, modelos matemáticos resolvidos por algoritmos exatos mostram-se aplicáveis na prática. Nesses contextos de instâncias reduzidas, os métodos exatos conseguem processar as informações e retornar a solução ótima, propiciando o melhor agrupamento de pedidos e rotas de separação mais curtas em um tempo computacional viável para a operação.

Entretanto, a realidade de grandes centros de distribuição voltados para o comércio eletrônico (*e-commerce*), a exemplo de operações logísticas em larga escala como as do Mercado Livre (Mercado Livre, 2025), impõe desafios complexos. Nesses ambientes, processam-se milhares de pedidos em curtos intervalos de tempo (Yang; Zhao; Guo, 2020). Diante desse volume, a complexidade combinatória inerente ao Problema da Seleção de Pedidos manifesta-se diretamente na operação diária.

Neste cenário de larga escala, o uso exclusivo de métodos exatos torna-se restrito. A explosão combinatória eleva o tempo computacional necessário para atestar a otimalidade de uma solução a patamares superiores às janelas operacionais de despacho. Em um mercado onde as entregas expressas são requisitos logísticos, aguardar o término do processamento de um algoritmo exato pode inviabilizar o cumprimento das metas de expedição.

Assim, a motivação prática deste trabalho reside na necessidade de algoritmos capazes de formar ondas de coleta com baixa latência temporal. O objetivo operacional é obter soluções de boa qualidade, que reduzam a quantidade de corredores visitados, dentro de limites de tempo compatíveis com o fluxo do armazém. O uso de heurísticas e metaheurísticas justifica-se pela sua escalabilidade, buscando propiciar redução de custos operacionais tanto para pequenos estoques quanto para centros de distribuição de alto volume.

1.1.2 Motivação Acadêmica

Do ponto de vista acadêmico, o PSPO apresenta desafios combinatórios relevantes. Sua classificação como *NP-Hard* inviabiliza a aplicação de métodos exatos para instâncias de alta dimensionalidade dentro de janelas operacionais restritas. Adicionalmente, o caráter fracionário da função objetivo altera o comportamento de convergência dos algoritmos tradicionais, incentivando a investigação de heurísticas orientadas ao denominador da função e o desenvolvimento de estruturas de avaliação incremental de estoques com complexidade temporal em $O(k)$, contribuindo para a literatura de Pesquisa Operacional aplicada à intralogística.

1.2 Objetivos

Esta seção detalha o objetivo geral e os objetivos específicos que direcionam a condução da pesquisa.

1.2.1 Objetivo Geral

O objetivo geral desta monografia é investigar, implementar e avaliar métodos de otimização combinatória, com ênfase em heurísticas e metaheurísticas reativas, para a resolução do Problema da Seleção de Pedidos Ótima (PSPO). Busca-se estruturar uma arquitetura computacional de alto desempenho capaz de encontrar soluções de elevada produtividade na razão itens por corredor, assegurando rigor temporal competitivo em relação a métodos exatos e ao estado da arte da literatura.

1.2.2 Objetivos Específicos

Para o alcance do objetivo geral delineado, definiram-se os seguintes objetivos específicos:

- Realizar revisão bibliográfica sobre os problemas de *Order Picking* e *Order Batching*, bem como sobre as heurísticas construtivas, estratégias de busca local multi-vizinhança e metaheurísticas aplicadas à logística de armazenagem;
- Formalizar matematicamente o PSPO, caracterizando as restrições operacionais de capacidade de onda (*wave boundaries*), os balanços dinâmicos de estoque e a não linearidade da função objetivo fracionária;
- Conceber e implementar o *OOSP-MetaSolver Framework* na linguagem Rust, integrando avaliação incremental de deltas em tempo $O(k)$, heurísticas construtivas orientadas por corredores (*Aisle-Centric*), exploração multi-vizinhança estruturada via *Variable Neighborhood Descent* (VND) e as metaheurísticas GRASP Reativo (R-GRASP) e AR-BPSO com paralelismo nativo;

- Conduzir protocolo experimental sistemático sobre o conjunto oficial de 50 instâncias reais do desafio Mercado Livre/SBPO 2025 sob o limite temporal fixo de 600 segundos;
- Analisar comparativamente o desempenho das abordagens desenvolvidas frente a *solvers* exatos de programação matemática e à literatura recente, avaliando o compromisso (*trade-off*) entre tempo computacional, consistência convergencial e qualidade da solução primal.

1.3 Organização do Trabalho

O documento estrutura-se em seis capítulos interconectados, organizados progressivamente desde a fundamentação teórica até a consolidação empírica e conclusiva dos resultados:

Capítulo 2: Revisão Bibliográfica. Consolida o referencial teórico sobre otimização combinatória, intralogística de centros de distribuição, problemas de *Order Picking* e *Order Batching*, além de revisar criticamente os fundamentos de heurísticas de busca local e metaheurísticas de trajetória e populacionais aplicadas à área.

Capítulo 3: Definição do Problema. Detalha formalmente o Problema da Seleção de Pedidos Ótima (PSPO). Apresenta a formulação matemática integral do modelo de Programação Fracionária, descreve as restrições operacionais de estoque e capacidade e examina a estrutura computacional dos conjuntos de instâncias A, B e X do desafio Mercado Livre/SBPO 2025.

Capítulo 4: Metodologia e Desenvolvimento. Apresenta a concepção e os detalhes arquiteturais do *OOSP-MetaSolver Framework* em Rust. Explica a representação binária compacta via *bitsets*, o avaliador de variações em tempo constante ou linear em relação ao tamanho da vizinhança ($O(k)$), os algoritmos construtivos orientados a pedidos e a corredores, o procedimento de refinamento local VND multi-vizinhança e as formulações das metaheurísticas R-GRASP e AR-BPSO.

Capítulo 5: Resultados Computacionais. Dedica-se à apresentação, análise e discussão dos experimentos computacionais realizados nas 50 instâncias oficiais do problema. Examina a evolução do ganho de produtividade ao longo das etapas do *pipeline* algorítmico, compara as metaheurísticas propostas e confronta os desempenhos alcançados perante modelos exatos e publicações da literatura científica.

Capítulo 6: Considerações Finais. Apresenta as considerações finais da monografia, sintetizando os principais achados teóricos e empíricos, atestando o cumprimento dos objetivos gerais e específicos, avaliando implicações práticas e acadêmicas, bem como delimitando as limitações metodológicas e propondo linhas de investigação para trabalhos futuros.

2 Revisão Bibliográfica

Este capítulo apresenta a base conceitual e o estado da arte que fundamentam esta pesquisa. O capítulo está dividido em duas seções principais: a primeira, Fundamentação Teórica, detalha os conceitos de otimização combinatória, heurísticas e define o *Order Batching Problem* (OBP). A segunda seção, Trabalhos Relacionados, analisa e compara as principais publicações científicas da área com a proposta de abordagem desta monografia, situando a contribuição deste trabalho.

2.1 Fundamentação Teórica

Esta seção apresenta os conceitos teóricos fundamentais para a compreensão deste trabalho, com foco em problemas de otimização combinatória e nas técnicas de Inteligência Computacional usadas para resolvê-los. Serão definidas as classes de heurísticas e as metaheurísticas que formam a base das metodologias de otimização aplicadas ao *Order Batching Problem* (OBP). As definições e explicações a seguir são baseadas na referência (Souza, 2024).

2.1.1 Problemas de Otimização

Muitos problemas práticos de otimização podem ser modelados da seguinte forma: dado um conjunto S de variáveis discretas s, denominadas soluções, e uma função objetivo $f : S \to \mathbb{R}$ que associa cada solução s a um valor real $f(s)$, o objetivo é encontrar a solução ótima $s^* \in S$, para a qual $f(s)$ tem o valor mais favorável, seja ele mínimo ou máximo.

Grande parte desses problemas, incluindo o *Order Batching Problem* (OBP), são de natureza combinatória e classificados como *NP-hard*. Isso significa que ainda não existem algoritmos capazes de resolvê-los para o ótimo em tempo polinomial. A dificuldade reside no tamanho do espaço de busca, ou seja, o número de soluções possíveis para o problema.

Embora métodos exatos, como *branch-and-bound* (Land; Doig, 1960), possam reduzir o número de soluções analisadas, o tempo para encontrar a solução ótima ainda pode ser proibitivo. Na prática, é suficiente encontrar uma boa solução em um custo computacional aceitável. Por essa razão, pesquisadores têm focado no uso de heurísticas. Uma *heurística* é definida como uma "técnica inspirada em processos intuitivos que procura uma boa solução [...] sem, no entanto, estar capacitada a garantir sua otimalidade"(Souza, 2024, p. 1-2).

2.1.2 Heurísticas Construtivas

Uma heurística construtiva tem como objetivo construir uma solução para o problema, adicionando um elemento de cada vez. Nas heurísticas clássicas, os elementos candidatos à

inserção são geralmente ordenados por uma função gulosa (*greedy*), que estima o benefício de inserir cada elemento, e somente o melhor elemento é inserido a cada passo. Essa função gulosa geralmente está associada à contribuição na função objetivo que a inserção daquele elemento na solução causa.

Uma alternativa muito comum é a construção aleatória, onde, a cada passo, o elemento a ser inserido na solução é selecionado aleatoriamente dentre os candidatos. A vantagem é a simplicidade de implementação, mas a desvantagem é a baixa qualidade média da solução final, exigindo maior esforço na fase de refinamento. Muitas metaheurísticas, como o *Greedy Randomized Adaptive Search Procedure* (GRASP) (Resende; Ribeiro, 2010), utilizam uma combinação dessas duas abordagens.

2.1.3 Heurísticas de Busca Local (Refinamento)

Heurísticas de refinamento, também chamadas de técnicas de busca local, partem de uma solução inicial (gerada por uma heurística construtiva ou aleatoriamente) e tentam melhorá-la. Elas se baseiam no conceito de vizinhança.

Para uma solução s do espaço de busca S, uma função de vizinhança N associa s a um conjunto $N(s) \subseteq S$, onde cada solução $s' \in N(s)$ é chamada de vizinha de s. Um movimento m é a modificação que transforma s em s'. O método move-se de vizinho em vizinho, geralmente aceitando apenas movimentos que melhorem a função objetivo. O principal problema das heurísticas de busca local é que elas param quando encontram um ótimo local, que é uma solução onde nenhum de seus vizinhos é melhor, mas que não é necessariamente a melhor solução global do problema.

2.1.4 Metaheurísticas

Para evitar que a busca fique presa em ótimos locais, utilizam-se as metaheurísticas. Elas são procedimentos de caráter geral que controlam uma heurística subordinada, possuindo mecanismos para escapar de ótimos locais e explorar o espaço de busca de forma mais ampla, sendo diferenciadas, principalmente, pela escolha desse mecanismo.

As metaheurísticas podem ser divididas em duas categorias principais (Gendreau; Potvin, 2010a):

- **Busca Local ou Trajetória:** a exploração é feita aplicando movimentos sobre uma única solução corrente. Exemplos incluem a Busca Tabu (TS) (Gendreau; Potvin, 2010b) e o *Greedy Randomized Adaptive Search Procedure* (GRASP) (Resende; Ribeiro, 2010).
- **Busca Populacional:** um conjunto denominado população de boas soluções é mantido e combinado para tentar produzir soluções ainda melhores. Exemplos incluem a *Particle*

Swarm Optimization (PSO) (Kennedy; Eberhart, 1995) e os *Genetic Algorithms* (GA) (Reeves, 2010).

Considerando que o presente trabalho desenvolve arquiteturas baseadas em *Greedy Randomized Adaptive Search Procedure* (GRASP) e *Particle Swarm Optimization* (PSO), além de utilizar a Busca Tabu (TS) como um dos métodos de referência para a avaliação de desempenho das estruturas de busca local propostas, o funcionamento destas metaheurísticas é detalhado a seguir.

2.1.4.1 Busca Tabu

A Busca Tabu (TS) é uma metaheurística de busca local formalizada e consolidada nos trabalhos de Glover e Laguna (1997), sendo amplamente revisada na literatura de otimização (Gendreau; Potvin, 2010b). É um método que explora o espaço de soluções movendo-se de uma solução para o seu melhor vizinho, mesmo que esse vizinho seja pior que a solução corrente, caracterizando um movimento de piora. Esta estratégia permite que o método escape de ótimos locais.

Para evitar que o algoritmo retorne imediatamente à solução anterior e entre em um ciclo infinito, a TS utiliza uma estrutura de memória chamada lista tabu T. Esta lista armazena informações sobre os últimos $|T|$ movimentos realizados, onde $|T|$ é um parâmetro, geralmente proibindo seus movimentos reversos para evitar que o algoritmo retorne a soluções já visitadas. Ocasionalmente, um movimento tabu pode ser aceito se ele levar a uma solução melhor que a melhor solução encontrada até o momento s^*. Esse mecanismo é chamado de função de aspiração. A função de aspiração é gerenciada por uma memória auxiliar, aqui denotada por A.

O Algoritmo 2.1 apresenta o pseudocódigo da Busca Tabu para um problema de maximização.

Algoritmo 2.1: Pseudocódigo do algoritmo de Busca Tabu.

Input: $f(\cdot), N(\cdot), A(\cdot), |V|, |T|, BTmax, s$
Output: Melhor solução encontrada s^*

1 $s^* \leftarrow s$
2 $Iter \leftarrow 0$
3 $MelhorIter \leftarrow 0$
4 $T \leftarrow \emptyset$
5 Inicialize A
6 **while** $(Iter - MelhorIter \leq BTmax)$ **do**
7 $Iter \leftarrow Iter + 1$
8 Selecione $s' \in N(s)$ tal que seja o melhor vizinho não tabu ou atenda critério de aspiração
9 Atualize T
10 $s \leftarrow s'$
11 **if** $f(s) > f(s^*)$ **then**
12 $s^* \leftarrow s$
13 $MelhorIter \leftarrow Iter$
14 **end**
15 Atualize A
16 **end**
17 **return** s^*

O procedimento inicia definindo a solução corrente s como a melhor solução encontrada s^* (Linha 1). Um conjunto de iterações é executado até que um critério de parada seja atingido. Neste algoritmo, o critério adotado é o número máximo de iterações consecutivas sem melhoria na melhor solução global (s^*), denotado por $BTmax$ (Linha 6). Note que este critério difere do limite de iterações totais, pois o contador de estagnação é reiniciado sempre que uma nova melhor solução é encontrada. A cada passo, o algoritmo explora a vizinhança $N(s)$ e seleciona o melhor movimento que não esteja na lista tabu T ou que satisfaça um critério de aspiração A (Linha 8). A lista tabu é atualizada para impedir o retorno a soluções recentemente visitadas (Linha 9), permitindo que a busca escape de ótimos locais. Se a nova solução s' for melhor que a melhor global s^* (Linha 11), esta é atualizada (Linha 12).

Para uma revisão mais aprofundada sobre esta metaheurística, consultar o trabalho de Gendreau e Potvin (2010a).

2.1.4.2 *Greedy Randomized Adaptive Search Procedure* (GRASP)

O *Greedy Randomized Adaptive Search Procedure* (GRASP), formalizado inicialmente por Feo e Resende (1995) e amplamente revisado na literatura de otimização (Resende; Ribeiro, 2010), é uma metaheurística multi-partida iterativa. Cada iteração do GRASP consiste em duas

fases principais: uma fase de construção, onde uma solução inicial viável é produzida, e uma fase de busca local (ou refinamento), na qual a vizinhança dessa solução inicial é explorada em busca de um ótimo local.

Na fase de construção, a cada passo de inserção de um elemento na solução, é construída uma Lista Restrita de Candidatos (RCL, *Restricted Candidate List*), formada pelos melhores elementos disponíveis de acordo com uma função gulosa. Um elemento da RCL é então selecionado aleatoriamente para compor a solução. O grau de aleatoriedade e de ganância é controlado pelo parâmetro $\alpha \in [0, 1]$; quando $\alpha = 0$, o procedimento é puramente guloso, enquanto $\alpha = 1$ representa uma escolha puramente aleatória. Em seguida, a fase de busca local aplica sucessivos movimentos sobre a solução construída até alcançar um ótimo local. O processo repete-se por um número pré-determinado de iterações ou até atingir um critério de parada, e a melhor solução global encontrada ao longo de todas as iterações é retornada como resultado final.

O Algoritmo 2.2 apresenta o pseudocódigo do funcionamento geral da metaheurística GRASP para um problema de maximização.

Algoritmo 2.2: Pseudocódigo do algoritmo GRASP.

Input: $f(\cdot), N(\cdot), \alpha, MaxIter$
Output: Melhor solução encontrada s^*

```
1  Iter ← 0
2  s* ← ∅
3  while (Iter < MaxIter) do
4      Iter ← Iter + 1
5      s ← ConstrucaoGulosoAleatoria(α)
6      s ← BuscaLocal(s, N)
7      if s* = ∅ ou f(s) > f(s*) then
8          s* ← s
9      end
10 end
11 return s*
```

O procedimento inicia com a definição do contador de iterações e a inicialização da melhor solução global s^* (Linha 2). O algoritmo executa um ciclo repetitivo controlado pelo critério de parada, comumente o número máximo de iterações $MaxIter$ (Linha 3). Em cada iteração, uma solução inicial s é gerada pelo procedimento de construção gulosa e aleatorizada regulado pelo parâmetro α (Linha 5). Na sequência, a fase de busca local explora a vizinhança $N(s)$ para refinar a solução até um ótimo local (Linha 6). Caso a nova solução refinada s supere a qualidade da melhor solução global até então conhecida s^* (Linha 7), esta é atualizada (Linha 8).

Para uma revisão mais aprofundada sobre esta metaheurística e suas variações reativas e híbridas, consultar o trabalho de Resende e Ribeiro (2010).

2.1.4.3 *Particle Swarm Optimization* (PSO)

A *Particle Swarm Optimization* (PSO) é uma metaheurística populacional proposta originalmente por Kennedy e Eberhart (1995), inspirada no comportamento coletivo de enxames e bandos de animais, como aves ou cardumes. No PSO, uma população de soluções candidatas, denominadas partículas, desloca-se pelo espaço de busca multidimensional para encontrar a solução ótima. Cada partícula possui uma posição, que representa uma solução para o problema, e uma velocidade, que dita a direção e a magnitude de seu movimento a cada geração.

O movimento de cada partícula é influenciado por dois fatores de aprendizado: sua própria experiência individual, representada pela melhor posição já visitada pela partícula até o momento ($pbest$), e a experiência social do enxame, representada pela melhor posição encontrada por qualquer partícula da população ($gbest$). A cada geração, o vetor de velocidade de cada partícula é ajustado em direção a $pbest$ e $gbest$, ponderado por coeficientes de aceleração e por uma inércia que equilibra a exploração global do espaço com a exploração local das melhores regiões encontradas.

O Algoritmo 2.3 apresenta o pseudocódigo do funcionamento geral do algoritmo PSO canonizado para otimização.

Algoritmo 2.3: Pseudocódigo do algoritmo de Otimização por Enxame de Partículas (PSO).

Input: $f(\cdot), M, MaxGen, w, c_1, c_2$

Output: Melhor solução global encontrada $gbest$

1 Inicialize uma população de M partículas com posições x_i e velocidades v_i aleatórias
2 **forall** *partícula* $i \in \{1, \ldots, M\}$ **do**
3 $\quad pbest_i \leftarrow x_i$
4 **end**
5 $gbest \leftarrow \arg\max_{x_i} f(x_i)$
6 $Gen \leftarrow 0$
7 **while** $(Gen < MaxGen)$ **do**
8 $\quad Gen \leftarrow Gen + 1$
9 $\quad$ **forall** *partícula* $i \in \{1, \ldots, M\}$ **do**
10 $\quad\quad$ Calcule a nova velocidade
$\quad\quad\quad v_i \leftarrow w \cdot v_i + c_1 \cdot r_1 \cdot (pbest_i - x_i) + c_2 \cdot r_2 \cdot (gbest - x_i)$
11 $\quad\quad$ Atualize a posição $x_i \leftarrow x_i + v_i$
12 $\quad\quad$ **if** $f(x_i) > f(pbest_i)$ **then**
13 $\quad\quad\quad pbest_i \leftarrow x_i$
14 $\quad\quad\quad$ **if** $f(pbest_i) > f(gbest)$ **then**
15 $\quad\quad\quad\quad gbest \leftarrow pbest_i$
16 $\quad\quad\quad$ **end**
17 $\quad\quad$ **end**
18 $\quad$ **end**
19 **end**
20 **return** $gbest$

O procedimento inicia com a geração de uma nuvem com M partículas, atribuindo-lhes posições x_i e velocidades v_i aleatórias no espaço de busca (Linha 1). Cada partícula inicializa sua memória cognitiva individual $pbest_i$ na sua posição inicial (Linha 3), e a melhor posição inicial dentre todas as partículas é designada como a memória social global $gbest$ (Linha 5). O laço evolutivo repete-se até o critério de parada ser atingido, como o número máximo de gerações $MaxGen$ (Linha 7). Em cada geração, para todas as partículas do enxame (Linha 9), a velocidade v_i é atualizada (Linha 10) combinando a inércia (w), a atração cognitiva em direção a $pbest_i$ controlada pelo coeficiente c_1 e pelo número aleatório $r_1 \sim U(0, 1)$, e a atração social em direção a $gbest$ controlada por c_2 e $r_2 \sim U(0, 1)$. A posição da partícula é então atualizada com base na nova velocidade (Linha 11). Caso a nova posição avaliada resulte em uma função objetivo superior à sua melhor experiência individual (Linha 12), o $pbest_i$ é atualizado (Linha 13). Se essa melhoria individual também superar a melhor solução do enxame (Linha 14), o $gbest$ global é atualizado para refletir o novo ótimo encontrado (Linha 15).

Para uma revisão mais aprofundada sobre a metaheurística PSO e suas formulações discretas, consultar os trabalhos de Kennedy e Eberhart (1995) e Kennedy e Eberhart (1997).

2.1.5 Configuração Automática de Algoritmos

O desempenho de algoritmos heurísticos e metaheurísticos é altamente sensível à configuração de seus parâmetros, como taxas de mutação, tamanhos de listas de memória ou critérios de aceitação. Tradicionalmente, a calibração desses parâmetros era feita de forma manual, baseada em tentativa e erro ou em um *grid search* limitado, o que consome tempo considerável e não garante a identificação da configuração ótima para o problema em questão.

Para superar essas limitações, surgiram as técnicas de Configuração Automática de Algoritmos (AAC, *Automatic Algorithm Configuration*). O objetivo da AAC é encontrar automaticamente o conjunto de parâmetros que maximiza o desempenho do algoritmo em um conjunto representativo de instâncias de treino. Dentre as ferramentas mais difundidas na literatura para este fim, destaca-se o pacote *irace* (López-Ibáñez *et al.*, 2016).

O *irace* baseia-se no conceito de *Iterated Racing*. O algoritmo inicia com uma amostragem de diversas configurações candidatas do espaço de parâmetros. Essas configurações são então submetidas a uma "corrida"(*racing*), onde são testadas em instâncias do problema. À medida que evidências estatísticas (através de testes como o de Friedman ou o *t-test*) demonstram que certas configurações são significativamente piores que outras, estas são descartadas para economizar tempo computacional. Ao final de uma iteração (ou corrida), o modelo de amostragem é atualizado com base nas melhores configurações sobreviventes (*elites*), concentrando a busca em regiões mais promissoras do espaço de parâmetros nas iterações subsequentes.

2.2 Trabalhos Relacionados

Esta seção apresenta uma análise dos principais trabalhos da literatura que abordam o *Order Batching Problem* (OBP) e problemas correlatos. O objetivo é descrever os métodos, objetivos e resultados alcançados por outros pesquisadores, comparando-os com a formulação do problema desta monografia. Esta análise permite identificar o estado da arte e posicionar a contribuição deste trabalho.

2.2.1 Panorama de Abordagens para o Order Batching

A otimização do agrupamento de pedidos (*Order Batching Problem*, OBP) tem sido amplamente investigada na Pesquisa Operacional nas últimas décadas. O problema é frequentemente abordado através de métodos que buscam otimizar atividades-chave, como o agrupamento, responsável por definir o lote, o roteamento, que define o percurso, e o sequenciamento, focado em organizar a ordem de coleta (Pardo *et al.*, 2024).

A literatura classifica os métodos de resolução em três grandes categorias, dependendo da complexidade da instância e das restrições consideradas:

- **Métodos Exatos:** abordagens como Programação Dinâmica, *Branch-and-Bound* e formulações de Programação Linear Inteira Mista (PLIM) são utilizadas para garantir a otimalidade da solução. No entanto, devido à natureza *NP-hard* do problema, esses métodos são geralmente restritos a instâncias de pequeno a médio porte ou requerem tempos computacionais elevados (Pardo *et al.*, 2024). Recentemente, Santos e Baldotto (2025) propuseram formulações exatas baseadas em reformulação linear para o problema específico do desafio proposto pelo Mercado Livre no Simpósio Brasileiro de Pesquisa Operacional (SBPO) (Mercado Livre, 2025), conseguindo resolver diversas instâncias até a otimalidade.
- **Heurísticas e Metaheurísticas:** para instâncias de grande porte, comuns em cenários reais de *e-commerce*, predominam as abordagens heurísticas. Pardo *et al.* (2024) destacam o uso extensivo de metaheurísticas de busca local e populacionais. Técnicas como *Simulated Annealing* (SA), *Iterated Local Search* (ILS), Busca Tabu (TS) e *Genetic Algorithms* (GA) são aplicadas para explorar o espaço de soluções de forma eficiente, buscando um equilíbrio entre qualidade da solução e tempo de execução (Pardo *et al.*, 2024).
- **Abordagens Híbridas:** uma tendência atual é a combinação de métodos exatos com heurísticas para aproveitar os pontos fortes de ambos. O *framework Generate-and-Solve*, utilizado por Saraiva *et al.* (2025), exemplifica essa classe, onde uma metaheurística gera subconjuntos de soluções promissoras que são subsequentemente otimizadas por um modelo matemático exato.

Este trabalho se insere no contexto das metaheurísticas, investigando algoritmos capazes de fornecer soluções competitivas para as instâncias desafiadoras onde métodos exatos tradicionais podem falhar em tempo hábil.

2.2.2 Otimização de Coleta de Pedidos em Lote em Armazéns de *E-commerce* (Yang; Zhao; Guo, 2020)

Yang, Zhao e Guo (2020) abordam a otimização da coleta de pedidos em lote, denominada *order batch picking*, em armazéns de *e-commerce*, focando especificamente em sistemas de armazenamento *multi-local*, nos cenários 1-n e n-n. A justificativa central é que, nesses sistemas, a decisão de qual localização escolher para coletar um item (SKU) que existe em múltiplos pontos afeta drasticamente a distância total percorrida, sendo um subproblema de otimização pouco explorado. O objetivo do artigo é minimizar a distância total de coleta, formulando modelos matemáticos para os três tipos de armazenamento (1-1, 1-n e n-n) e propondo um pacote de algoritmos aninhados, envolvendo seleção de localização, roteamento e *batching*, para resolvê-los. Para o problema de *batching*, os autores utilizam *Branch-and-Price* (BPB) para instâncias

pequenas e um algoritmo de Busca Tabu para instâncias de grande escala. Os resultados dos experimentos numéricos mostraram que os algoritmos propostos foram satisfatórios tanto em qualidade de solução quanto em eficiência computacional. A conclusão principal dos autores é que, para problemas de média e grande escala, a combinação do algoritmo de seleção de localização LSA3 (*Minimum aisle coverage*) com o TSB (*Tabu Search Batching*) é a mais prática e eficiente. A diferenciação no nome (TSB) é relevante no contexto do trabalho original, pois os autores também propõem uma variante da Busca Tabu especificamente para a seleção de localizações (LSA7).

O trabalho é altamente relevante, pois trata do mesmo problema central desta monografia: o *Order Batching Problem* (OBP). Contudo, as propostas de abordagens diferem fundamentalmente no objetivo e na abstração. Yang, Zhao e Guo (2020) buscam minimizar a distância de viagem exata, o que requer algoritmos de roteamento explícitos em nível de localização. O problema desta monografia busca maximizar a produtividade através da relação de itens por corredor, usando o número de corredores como um *proxy* para a distância, sem modelar o roteamento. Além disso, as restrições são distintas: esta pesquisa foca em limites de unidades totais na *wave* (LB/UB) e estoque no corredor, enquanto Yang, Zhao e Guo (2020) focam na capacidade em número de pedidos. A contribuição desta monografia será, portanto, aplicar técnicas de otimização para esta formulação específica que visa concentrar a coleta no menor número de corredores.

2.2.3 Otimização Conjunta de *Batching*, Sequenciamento e Roteamento (Coruzzolo *et al.*, 2023)

Coruzzolo *et al.* (2023) propõem um modelo para a otimização conjunta de três problemas correlacionados: o *Order Batching*, o *Batch Assignment Sequencing*, referente ao sequenciamento dos lotes para os coletores, e o *Picking Routing*, que trata do roteamento da coleta. A justificativa é que tratar esses problemas de forma independente, como é comum na literatura, leva a soluções sub-ótimas. O principal objetivo do modelo é minimizar o tempo total de conclusão das coletas e, crucialmente, minimizar a soma ponderada de atrasos, denominada *tardiness*, e adiantamentos, denominados *earliness*, dos pedidos em relação aos seus prazos de entrega, os *due dates*. Devido à alta complexidade do modelo de programação linear (MILP) proposto, a solução foca em uma heurística construtiva, denominada *Earliest Due Date* (EDD), melhorada por algoritmos de *Iterated Local Search* (ILS). Os resultados de um estudo de caso real mostraram que a estratégia de *batching* proposta reduziu o tempo total de *picking* em 57% em comparação com a estratégia de pedido único. A conclusão dos autores é que os algoritmos baseados em ILS foram capazes de identificar com sucesso a solução mínima e melhorar significativamente a solução inicial obtida pela heurística EDD.

Este trabalho é relevante por também focar no *Order Batching Problem* (OBP), mas sua proposta de abordagem é significativamente mais complexa que a proposta desta monografia. A

principal diferença é o escopo: Coruzzolo *et al.* (2023) resolvem explicitamente o sequenciamento e o roteamento, um problema do tipo *Travelling Salesman Problem* (TSP) (Applegate *et al.*, 2006), em nível de localização. O problema desta monografia não modela o roteamento nem o sequenciamento, mas abstrai a eficiência da coleta através da seleção de corredores. Além disso, os objetivos divergem: o artigo foca em minimizar o tempo e as penalidades por atraso, a *tardiness*, enquanto esta monografia foca em maximizar a produtividade. As restrições de *due dates*, centrais para Coruzzolo *et al.* (2023), não são o foco do problema tratado nesse trabalho, que se concentra em limites de unidades (LB/UB) e disponibilidade de estoque.

2.2.4 Formulação Linear e Algoritmo Exato para o PSPO (Santos; Baldotto, 2025)

Santos e Baldotto (2025) abordam diretamente o Problema da Seleção de Pedidos Ótima (PSPO), conforme proposto pelo Mercado Livre para o LVII SBPO (Mercado Livre, 2025). A justificativa do trabalho é a necessidade de desenvolver métodos exatos para resolver esta formulação de problema, que é classificada como um Problema de Programação Linear Inteira Mista Fracional (MILFP) *NP-hard*. O objetivo é criar e avaliar abordagens exatas para maximizar a métrica de produtividade, definida pelo total de itens coletados dividido pelo número de corredores visitados, respeitando as restrições de limites de itens da *wave* (LB/UB) e a disponibilidade de estoque. Os autores propõem três métodos principais: (1) uma reformulação linear (MILP) do problema fracionário, chamada `ref-lin`; (2) um algoritmo iterativo paralelo, `par-it`, que resolve subproblemas de forma eficiente ao fixar o número de corredores; e (3) uma abordagem híbrida (`par-it + ref-lin`) que utiliza o `par-it` para reduzir o espaço de busca antes de aplicar o modelo MILP completo. Os resultados experimentais nas 35 instâncias públicas do desafio mostraram que as abordagens exatas, particularmente a híbrida e a `par-it` pura, foram capazes de encontrar e garantir a otimalidade em 27 instâncias dentro do tempo limite de 600 segundos. A conclusão é que os métodos exatos são viáveis e eficientes para resolver instâncias de tamanho considerável do PSPO.

Este trabalho apresenta alta relevância para esta monografia, visto que aborda precisamente o mesmo Problema da Seleção de Pedidos Ótima (PSPO). A formulação do problema utilizada pelos autores, incluindo a função objetivo fracionária que maximiza a produtividade e as restrições operacionais de limites de itens (LB/UB) e disponibilidade de estoque, é idêntica à adotada neste estudo. A diferença fundamental entre as pesquisas reside na metodologia de solução. Santos e Baldotto (2025) focam exclusivamente no desenvolvimento e aplicação de métodos exatos, como uma reformulação linear (MILP) e um algoritmo iterativo paralelo, com o objetivo de encontrar e provar a otimalidade das soluções. Em contrapartida, a contribuição desta monografia será complementar: enquanto o trabalho de Santos e Baldotto (2025) estabelece um *baseline* robusto para a solução ótima, esta pesquisa explorará a aplicação de metaheurísticas para o mesmo problema. O objetivo será avaliar se abordagens heurísticas conseguem encontrar

soluções de alta qualidade em um tempo computacional significativamente menor, ou se são capazes de obter melhores resultados para as instâncias complexas onde os métodos exatos não alcançaram a otimalidade dentro do tempo limite estipulado.

2.2.5 Generate-and-Solve aplicada ao Problema de Seleção de Pedidos (Saraiva *et al.*, 2025)

Saraiva *et al.* (2025) também abordam o Problema da Seleção de Pedidos Ótima (PSPO), no mesmo contexto do desafio SBPO. A justificativa do trabalho é a dificuldade *NP-hard* do problema, que motiva o uso de métodos que não sejam puramente exatos. Os autores propõem uma metodologia híbrida chamada *Generate-and-Solve* (GS), que combina uma metaheurística populacional, o *Biased Random-Key Genetic Algorithm* (BRKGA), com um método exato de Programação Linear Inteira. Nesta abordagem, o BRKGA atua como um Gerador de Instâncias Reduzidas (GIR), explorando e sugerindo subconjuntos promissores de corredores, enquanto um modelo de PLI atua como um Decodificador de Instâncias Reduzidas (DIR), resolvendo otimamente esses subproblemas menores. Os resultados computacionais em vinte instâncias mostraram que esta abordagem híbrida convergiu para a solução ótima, encontrada pelo método exato puro, em cinco casos. A conclusão dos autores é que, nos demais casos, os resultados foram muito próximos, demonstrando que o método GS é eficaz para reduzir a complexidade do problema e gerar soluções de alta qualidade em tempo hábil.

Este trabalho é de alta relevância, pois ataca o mesmo Problema da Seleção de Pedidos Ótima (PSPO) desta monografia, com a mesma função objetivo e restrições. A metodologia de Saraiva *et al.* (2025) é uma proposta de abordagem híbrida, que utiliza uma metaheurística (BRKGA) não para resolver o problema diretamente, mas para gerar subproblemas mais fáceis que são então resolvidos por um método exato (PLI). Esta proposta de abordagem se apresenta como uma alternativa interessante aos métodos puramente exatos, como o de Santos e Baldotto (2025). A análise deste artigo será fundamental para comparar os resultados de uma proposta de abordagem puramente heurística contra uma proposta de abordagem que combina o poder das heurísticas com a garantia de otimalidade dos métodos exatos em subproblemas.

2.2.6 Análise Comparativa dos Trabalhos Relacionados

A análise detalhada dos trabalhos relacionados revela diferentes propostas de abordagens para o *Order Batching Problem* (OBP) e problemas correlatos. Para facilitar a visualização das semelhanças e diferenças entre as metodologias, a Tabela 2.1 consolida as principais características de cada trabalho em comparação com a proposta desta monografia.

Tabela 2.1 – Tabela Comparativa Geral dos Trabalhos Relacionados.

Trabalho	**Problema Principal**	**Contexto**	**Função Objetivo**	**Métodos**	**Restrições Chave**
Yang et al. (2020) (Yang; Zhao; Guo, 2020)	Otimização conjunta de *batching*, seleção de localização e roteamento.	Armazéns de *e-commerce* com armazenamento multi-local.	Minimizar a distância total de viagem dos coletores.	Modelagem Matemática, Heurísticas, Metaheurística (Busca Tabu), Exato (*Branch-and-Price*).	Capacidade do lote, eliminação de sub-rota.
Coruzzolo et al. (2023) (Coruzzolo *et al.*, 2023)	Otimização conjunta de *batching*, sequenciamento e roteamento (JOBASRP).	Armazém *picker-to-part*.	Minimizar tempo de conclusão, *tardiness* e *earliness*.	MILP, Heurística (EDD), Metaheurística (*Iterated Local Search*).	Prazos de entrega, capacidade do veículo, roteamento explícito (TSP).
Santos e Baldotto (2025) (Santos; Baldotto, 2025)	Otimização exata do Problema da Seleção de Pedidos Ótima (PSPO).	Desafio de otimização SBPO 2025 (Mercado Livre).	Maximizar a produtividade.	Modelagem (MILFP, MILP), Reformulação Linear, Algoritmo Iterativo Paralelo.	Limites de tamanho da *wave*, disponibilidade de estoque por corredor.
Saraiva et al. (2025) (Saraiva *et al.*, 2025)	Otimização híbrida do PSPO (Metaheurística + Método Exato).	Desafio de otimização SBPO 2025 (Mercado Livre).	Maximizar a produtividade.	*Framework Generate-and-Solve*, BRKGA, PLI (OR-Tools).	Limites de tamanho da *wave*, disponibilidade de estoque por corredor.

Esta Proposta	Otimização metaheurística do Problema da Seleção de Pedidos Ótima (PSPO).	Armazéns de *e-commerce*.	Maximizar a produtividade.	Heurísticas Construtivas, Heurísticas de Busca Local, Metaheurísticas (GRASP e PSO).	Limites de tamanho da *wave*, disponibilidade de estoque por corredor.

3 Definição do Problema

Este capítulo apresenta a definição formal do problema abordado nesta monografia: o PSPO, conforme proposto no desafio da SBPO 2025 pelo Mercado Livre. Detalha-se a seguir a natureza fracionária do problema, a formulação matemática e o formato das instâncias. O foco desta pesquisa é especificamente na decisão estratégica de quais pedidos selecionar e quais corredores visitar para atendê-los, abstraindo o roteamento fino dos coletores dentro de cada corredor.

3.1 Otimização Fracionária e Seleção de Subconjuntos

O problema tratado nesta pesquisa difere de problemas clássicos de otimização linear onde se busca apenas minimizar custos ou maximizar lucros absolutos. Aqui, trata-se de um problema de Programação Fracionária (*Fractional Programming*).

O objetivo é otimizar uma razão que representa a eficiência ou produtividade da *wave*:

$$\text{Eficiência} = \frac{\text{Recurso Obtido}}{\text{Custo Incorrido}}$$

No contexto do desafio proposto, "Recurso Obtido" é o número total de itens coletados e "Custo Incorrido" é o número de corredores que precisam ser visitados para coletar esses itens. O problema de otimização combinatória resultante busca selecionar simultaneamente:

1. Um subconjunto de pedidos (O') da carteira de pedidos pendentes (*backlog*);
2. Um subconjunto de corredores (A') do armazém.

O objetivo é selecionar O' e A' tal que a razão entre os itens contidos nos pedidos de O' e o número de corredores em A' seja maximizada, respeitando as restrições de capacidade da *wave* e disponibilidade de estoque nos corredores. Essa natureza fracionária adiciona uma camada de complexidade significativa, pois a função objetivo não é linear.

A Figura 3.1 apresenta a estrutura de seleção de pedidos e a respectiva rota de coleta no armazém para uma carteira composta por 5 pedidos pendentes (p_0 a p_4), em que cada pedido especifica os itens exigidos e suas respectivas quantidades. No lado esquerdo da imagem, são selecionados para compor a *wave* os pedidos p_0, p_1 e p_3. O pedido p_0 demanda 4 itens (3 unidades do item 0 e 1 unidade do item 2); o pedido p_1 possui 2 itens (1 unidade do item 1 e 1 unidade do item 3); e o pedido p_3 totaliza 5 itens (1 unidade do item 0, 2 unidades do item 2, 1 unidade do item 3 e 1 unidade do item 4). O somatório da demanda desses três pedidos resulta em 11 itens. Como os limites de capacidade da *wave* impõem um mínimo de 5 itens ($LB = 5$) e um máximo

de 12 itens ($UB = 12$), conforme formalizado posteriormente pelas restrições (3.4) e (3.5), a demanda de 11 itens é factível.

No lado direito da Figura 3.1, ilustra-se o armazém estruturado em 5 corredores (a_0 a a_4), onde cada corredor contém os itens disponíveis no estoque e suas respectivas quantidades. Para atender aos 11 itens dos pedidos selecionados, opta-se pela visita aos corredores a_0 e a_1. A imagem destaca a marcação em ambos os corredores e apresenta a quantidade coletada de cada item neles: 4 unidades do item 0 (sendo 3 retiradas de a_0 e 1 de a_1), 1 unidade do item 1 (retirada de a_0), 3 unidades do item 2 (retiradas de a_1), 2 unidades do item 3 (retiradas de a_1) e 1 unidade do item 4 (retirada de a_1). Desse modo, a *wave* atende a 3 pedidos com 11 itens totais, exigindo a visita a 2 corredores. A razão entre o recurso obtido e o custo incorrido é de $11/2 = 5{,}5$ itens por corredor, valor correspondente à função objetivo (Z) do problema, expressa na Equação (3.1).

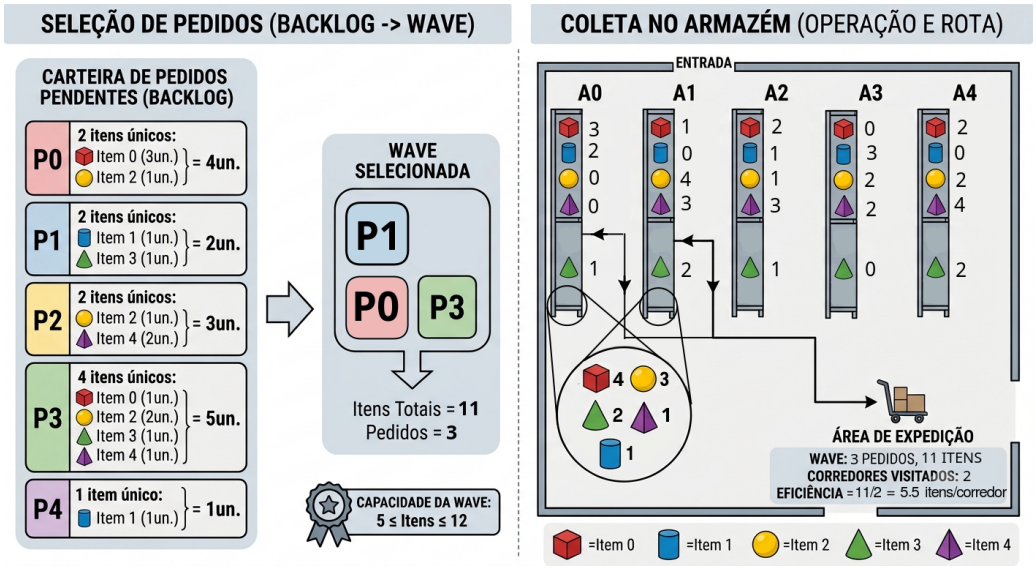

Figura 3.1 – Ilustração do processo de seleção de pedidos e corredores para uma *wave* com eficiência de 5,5 itens por corredor.

Fonte: Elaborado pelo autor.

Por sua vez, a Figura 3.2 ilustra uma solução alternativa para o mesmo armazém e a mesma carteira de pedidos. Nessa segunda configuração, são selecionados apenas os pedidos p_2 e p_3. O pedido p_2 demanda 3 itens (1 unidade do item 2 e 2 unidades do item 4), que, somados aos 5 itens do pedido p_3, totalizam 8 itens, respeitando os limites de capacidade permitidos ($5 \leq 8 \leq 12$). Nos corredores do armazém, seleciona-se apenas o corredor a_1, que possui em seu estoque todas as unidades necessárias para suprir a demanda desses dois pedidos. Desse modo, a expedição da *wave* atende a 2 pedidos com 8 itens totais ao custo da visita a 1 corredor (a_1), resultando na produtividade de $8/1 = 8$ itens por corredor.

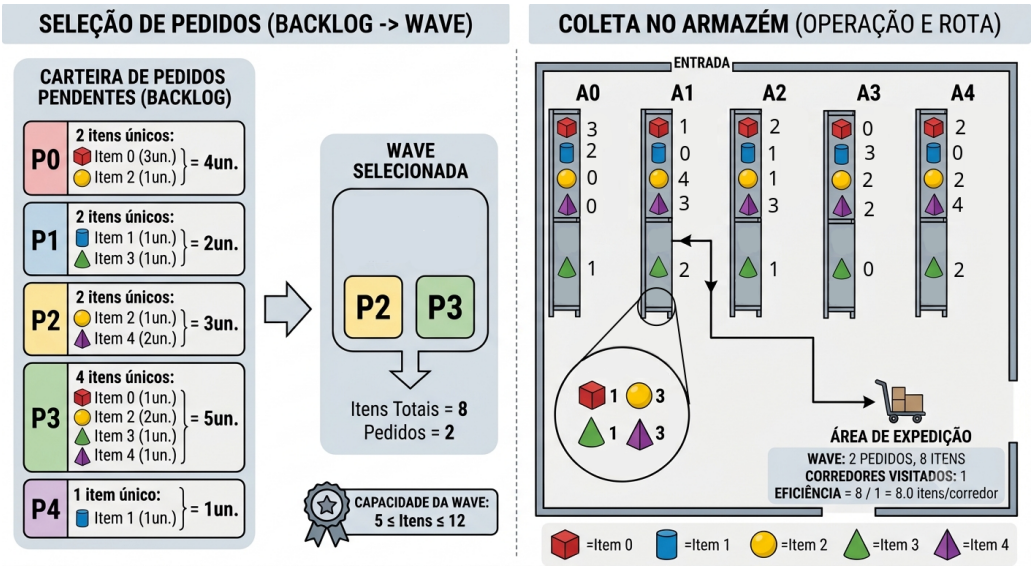

Figura 3.2 – Solução alternativa priorizando a eficiência da *wave*, atingindo a razão de 8 itens por corredor visitado.

Fonte: Elaborado pelo autor.

A análise comparativa entre as duas abordagens indica que a produtividade da *wave* na configuração da Figura 3.2 (8 itens por corredor) supera a obtida na Figura 3.1 (5,5 itens por corredor). Como a formulação matemática do problema (Equação 3.1) busca maximizar a razão entre itens coletados e corredores visitados, a segunda solução apresenta maior valor para a função objetivo. Esse comportamento demonstra que nem sempre a maximização absoluta do número de itens ou de pedidos coletados conduz à máxima eficiência da operação, evidenciando a natureza fracionária e não linear da otimização pretendida.

3.2 Formulação Matemática

Formalmente, o problema é modelado conforme descrito a seguir. Essa formulação foi proposta pelo Mercado Livre (2025) para o Desafio de Otimização do LVII Simpósio Brasileiro de Pesquisa Operacional.

3.2.1 Conjuntos e Parâmetros

Para representar o problema, considera-se um conjunto O de pedidos disponíveis $o \in \{0, \ldots, |O| - 1\}$ e um conjunto I de itens distintos ($i = 0, \ldots, |I| - 1$). O armazém é composto por um conjunto A de corredores ($a = 0, \ldots, |A| - 1$). A demanda do item i pelo pedido o é representada por d_{oi} e a disponibilidade de estoque do item i no corredor a é denotada por q_{ai}. Adicionalmente, são definidos os parâmetros LB e UB, que representam os limites inferior e superior de itens permitidos em uma *wave*, respectivamente.

3.2.2 Variáveis de Decisão

O modelo utiliza as seguintes variáveis de decisão: $x_o \in \{0, 1\}$ é uma variável binária que indica se o pedido o é selecionado para a *wave* (1) ou não (0); $y_a \in \{0, 1\}$ é uma variável binária que indica se o corredor a é visitado (1) ou não (0); e $w_{oai} \in \mathbb{Z}_+$ é uma variável inteira que representa a quantidade do item i do pedido o coletada no corredor a.

3.2.3 Modelo

Assim, o modelo tem como função objetivo, expressa pela Equação (3.1), maximizar a produtividade, definida pela razão entre o número de itens da *wave* e o número de corredores visitados.

$$\text{Maximizar } Z = \frac{\sum_{o \in O} \sum_{i \in I} d_{oi} \cdot x_o}{\sum_{a \in A} y_a} \tag{3.1}$$

Para que uma solução seja aceita, o modelo deve obedecer as restrições a seguir. A demanda de cada item de um pedido selecionado deve ser totalmente atendida pelos corredores escolhidos, como apresentado na equação (3.2).

$$\sum_{a \in A} w_{oai} = d_{oi} \cdot x_o, \quad \forall o \in O, \forall i \in I \tag{3.2}$$

A quantidade coletada não pode exceder o estoque disponível. Caso algum item seja coletado em um corredor, este deve ser marcado como visitado ($y_a = 1$), conforme a equação (3.3).

$$\sum_{o \in O} w_{oai} \leq q_{ai} \cdot y_a, \quad \forall a \in A, \forall i \in I \tag{3.3}$$

As restrições (3.4) e (3.5) garantem o respeito aos limites inferior e superior de itens na *wave*, respectivamente.

$$\sum_{o \in O} \sum_{i \in I} d_{oi} \cdot x_o \geq LB \tag{3.4}$$

$$\sum_{o \in O} \sum_{i \in I} d_{oi} \cdot x_o \leq UB \tag{3.5}$$

Por fim, define-se o domínio das variáveis de decisão:

$$x_o, y_a \in \{0, 1\}, \quad w_{oai} \in \mathbb{Z}_+ \tag{3.6}$$

3.3 Formato das Instâncias

Para a validação dos algoritmos, são utilizadas instâncias padronizadas do desafio.

3.3.1 Entrada

O arquivo de entrada (instância) é dividido em três blocos principais, com a seguinte estrutura de dados:

1. **Cabeçalho:** A primeira linha contém três valores inteiros: o número total de pedidos disponíveis ($|O|$), o número de tipos de itens no armazém ($|I|$), e o número total de corredores ($|A|$).
2. **Pedidos:** As $|O|$ linhas seguintes descrevem cada pedido. O formato é: a quantidade de itens únicos diferentes solicitados no pedido (N_{itens}), seguida por uma lista de pares (i, q_i), onde i é o índice do item (de 0 a $|I| - 1$) e q_i é a quantidade desse item.
3. **Corredores:** As $|A|$ linhas subsequentes descrevem o estoque disponível em cada corredor. O formato é: a quantidade de tipos de itens diferentes armazenados no corredor (M_{itens}), seguida por uma lista de pares (i, q_i), onde i é o índice do item e q_i é a quantidade estocada.
4. **Limites de Capacidade (*Wave*):** A última linha contém dois valores inteiros que representam o limite inferior (LB) e o limite superior (UB) para o total de unidades a serem coletadas na *wave* de pedidos selecionada.

O exemplo de entrada a seguir ilustra essa formatação, onde o problema envolve 5 pedidos, 5 itens e 5 corredores:

```
5 5 5             % |O|=5, |I|=5, |A|=5
2 0 3 2 1         % Pedido 0: 2 itens unicos. Item 0 (3 un.), Item 2 (1 un.)
2 1 1 3 1         % Pedido 1: 2 itens unicos. Item 1 (1 un.), Item 3 (1 un.)
2 2 1 4 2         % Pedido 2: 2 itens unicos. Item 2 (1 un.), Item 4 (2 un.)
4 0 1 2 2 3 1 4 1 % Pedido 3: 4 itens unicos. Item 0 (1 un.), Item 2 (2 un.), ...
1 1 1             % Pedido 4: 1 item unico. Item 1 (1 un.)
4 0 2 1 1 2 1 4 1 % Corredor 0: 4 itens unicos. Item 0 (2 un.), Item 1 (1 un.), ...
4 0 2 1 1 2 2 4 1 % Corredor 1: 4 itens unicos. Item 0 (2 un.), Item 1 (1 un.), ...
3 1 2 3 1 4 2     % Corredor 2: 3 itens unicos. Item 1 (2 un.), Item 3 (1 un.), ...
4 0 2 1 1 3 1 4 1 % Corredor 3: 4 itens unicos. Item 0 (2 un.), Item 1 (1 un.), ...
4 1 1 2 2 3 1 4 2 % Corredor 4: 4 itens unicos. Item 1 (1 un.), Item 2 (2 un.), ...
5 12              % Limites da Wave: LB=5, UB=12
```

3.3.2 Saída

A saída do algoritmo deve representar a solução ótima (ou uma solução de alta qualidade) e deve ser estruturada da seguinte forma, utilizando apenas índices de base zero:

1. A primeira linha informa a quantidade total de pedidos selecionados ($N_{pedidos}$).
2. As $N_{pedidos}$ linhas seguintes listam os índices dos pedidos que compõem a *wave*.
3. A linha subsequente (após o último índice de pedido) informa a quantidade total de corredores que precisam ser visitados ($N_{corredores}$).
4. As $N_{corredores}$ linhas finais listam os índices dos corredores que contêm pelo menos um item da *wave* de pedidos selecionada.

O exemplo a seguir ilustra uma possível solução para a instância de exemplo, onde os pedidos $0, 1, 2, 4$ são selecionados, resultando na visita aos corredores 1 e 3:

```
4  % Quantidade de Pedidos Selecionados (N_pedidos)
0  % Índice do Pedido 0
1  % Índice do Pedido 1
2  % Índice do Pedido 2
4  % Índice do Pedido 4
2  % Quantidade de Corredores Visitados (N_corredores)
1  % Índice do Corredor 1
3  % Índice do Corredor 3
```

4 Metodologia e Desenvolvimento

Este capítulo apresenta de forma detalhada o *framework* algorítmico desenvolvido para a resolução do Problema da Seleção de Pedidos Ótima (PSPO). Considerando a não linearidade da função objetivo fracionária, conforme modelado e formalizado no Capítulo 3 pela Equação 3.1, bem como as restrições operacionais de estoque, foi projetada a arquitetura de *software* denominada *Optimal Order Selection Problem Metaheuristic Solver Framework* (*OOSP-MetaSolver Framework*, Solucionador Metaheurístico para o Problema da Seleção de Pedidos Ótima). Serão descritas as representações computacionais utilizadas, os métodos de avaliação incremental, as heurísticas construtivas propostas, as abordagens de busca local e as duas metaheurísticas principais: o GRASP Reativo (R-GRASP) e o *Adaptive Reactive Binary Particle Swarm Optimization* (AR-BPSO).

4.1 Aspectos de Implementação e Desempenho Computacional

Para viabilizar a exploração do espaço de busca dentro do tempo limite de execução, a linguagem de programação Rust (edição 2024) foi adotada no desenvolvimento do *OOSP-MetaSolver Framework*. A escolha se fundamenta na capacidade da linguagem de fornecer segurança de memória (*memory safety*) em tempo de compilação e desempenho na execução de código nativo (Klabnik; Nichols, 2023). Por não depender de um coletor de lixo em tempo de execução (*Garbage Collector*), o sistema evita pausas imprevisíveis durante as iterações das metaheurísticas, mantendo ciclos de processamento uniformes e reduzindo a variabilidade do tempo de resposta.

Adicionalmente, a paralelização das avaliações populacionais e dos blocos iterativos foi implementada com a biblioteca *rayon* (versão 1.11) (Matsakis *et al.*, 2024). Essa abordagem possibilita o paralelismo em nível de dados (*data-parallelism*), distribuindo de forma assíncrona a carga de trabalho pelas múltiplas *threads* lógicas do processador. O balanceamento de carga automático da biblioteca por meio de rotinas de roubo de trabalho (*work-stealing*) (Blumofe *et al.*, 1995) otimiza a avaliação das estruturas de vizinhança sem incorrer em contenção excessiva de concorrência.

4.2 Representação Computacional e Avaliação Incremental

A eficiência da navegação efetuada pelas metaheurísticas é fortemente influenciada pela modelagem estrutural do problema alocado na memória do sistema.

4.2.1 Estruturas de Dados Densas e Representação Binária

Para permitir que o cálculo da função objetivo e as verificações de viabilidade ocorressem na escala de microssegundos, o PSPO foi mapeado para estruturas de dados básicas. As instâncias, originalmente modeladas de forma esparsa através de dicionários, são pré-processadas no momento da leitura para vetores densos, onde se armazena apenas o identificador do item e sua respectiva quantidade demandada. Adicionalmente, mapas estáticos de localização formados por vetores de bits contendo todos os corredores que armazenam um dado item, bem como uma matriz densa de estoque linearizada, são instanciados e compartilhados de forma somente-leitura ao longo de toda a execução das heurísticas.

Um componente essencial para a eficiência computacional é o pré-processamento do conjunto de corredores requeridos por pedido (*Order Required Aisles*). Para cada pedido pendente (do *backlog*), o *framework* calcula, no momento da leitura da instância, o subconjunto mínimo de corredores necessários para suprir integralmente sua demanda de itens, utilizando uma heurística de cobertura de conjuntos baseada em disponibilidade local. Esse cache estático elimina a necessidade de consultas repetitivas ao inventário global durante a fase de construção e busca local, transformando uma operação de busca complexa em um acesso direto à memória em tempo constante.

Uma solução candidata é representada de maneira compacta utilizando dois vetores binários, gerenciados através da estrutura *bitset*:

- *Vetor de Pedidos (x_o)*: Identifica, através de bits ativos, os pedidos selecionados para compor a *wave* de coleta.
- *Vetor de Corredores (y_a)*: Representa a infraestrutura ativada, indicando quais corredores físicos foram selecionados para visitação.

Essa representação permite realizar operações de união, interseção e diferença simétrica através de álgebra booleana intrínseca aos registradores da unidade central de processamento.

4.2.2 Avaliador de Balanço de Estoque Incremental

Uma das principais contribuições do *framework* para a eficiência computacional é o seu mecanismo de avaliação incremental de estoques, implementado através da estrutura *StockBalanceEvaluator*. Em uma implementação não incremental, a viabilidade do inventário é verificada por meio do recálculo do saldo de todos os itens após cada modificação na solução, o que resulta em complexidade de tempo $O(|I|)$.

A arquitetura proposta utiliza uma estratégia de avaliação baseada em deltas. O *StockBalanceEvaluator* mantém um vetor de estado que armazena a diferença entre o estoque total disponível nos corredores ativados e a demanda somada dos pedidos selecionados na solução cor-

rente. Ao realizar a inserção ou a remoção de um pedido, o algoritmo atualiza apenas as posições correspondentes aos itens afetados por essa operação. Essa abordagem reduz a complexidade da avaliação para $O(|I_o|)$, onde $|I_o|$ representa o número de itens distintos (*Stock Keeping Units*, SKUs) que compõem o pedido avaliado.

Para guiar as metaheurísticas de forma eficaz durante as transições de estado, o avaliador utiliza uma função de aptidão (*fitness*) baseada em penalidades dinâmicas. Embora o objetivo final seja a produtividade pura, a função interna aplicada durante a busca subtrai do valor objetivo uma penalidade proporcional ao desvio dos limites inferior (LB) ou superior (UB) da *wave*. Esse mecanismo induz a busca em direção à região de factibilidade, permitindo que o algoritmo explore soluções temporariamente inviáveis sem perder a convergência para o espaço de soluções válidas. A estratégia genérica de relaxamento de restrições por meio de funções de penalidade é amplamente utilizada na literatura de metaheurísticas (Gendreau; Potvin, 2010a). Contudo, a formulação específica adotada neste trabalho foi desenvolvida para representar as restrições particulares do Problema da Seleção Ótima de Pedidos, penalizando diretamente o desvio absoluto em relação aos limites inferior e superior da *wave*.

Adicionalmente, o avaliador implementa um mecanismo de baixa complexidade temporal para reversão de ações (*rollback*). Ao invés de realizar custosas cópias de vetores de alta dimensionalidade a cada movimento hipotético simulado pela busca local, o sistema adota uma abordagem baseada em registros de alteração (*change log*). Antes de modificar o saldo de qualquer item no vetor de estado, o identificador do item e seu saldo anterior são armazenados em uma pilha em memória. Caso um movimento avaliado acabe sendo rejeitado pelo algoritmo controlador, o avaliador simplesmente desempilha esse histórico, restaurando o valor prévio apenas dos saldos que foram efetivamente modificados. Essa operação de reversão restringe a sua complexidade de tempo à quantidade estrita de itens afetados, operando em $O(k)$, em que k representa o número de itens efetivamente modificados, o que favorece a exploração das vizinhanças em um tempo computacional viável.

4.2.3 Mecanismos de Sincronização e Reparo

Alterações estocásticas e isoladas efetuadas apenas nos vetores de pedidos ou exclusivas aos vetores de corredores geram frequentemente soluções inviáveis em decorrência do déficit natural de capacidade ou excesso indesejado de lotação da coleta. Para contornar este comportamento e evitar a exclusão imediata de boas candidatas, o *StockBalanceEvaluator* conta com heurísticas automáticas dedicadas à sincronização entre a demanda de pedidos e a infraestrutura de corredores ativada do modelo:

- *Sincronização de Corredores a partir de Pedidos*: Se a inserção imperativa de um novo pedido gerar saldos negativos, a sub-rotina busca iterativamente, fundamentada em critérios gulosos, os corredores ociosos que proporcionam a maior mitigação do atual déficit de itens,

ativando-os e embutindo-os na solução em cadeia até recuperar o estado de factibilidade total do sistema de distribuição.

- *Sincronização de Pedidos a partir de Corredores*: Caso o algoritmo heurístico venha a desativar um corredor por deliberação de custo ou eficiência, o déficit de itens imposto aos pedidos é solucionado rastreando e removendo os pedidos que se tornaram inválidos sem o estoque daquelas prateleiras.

Esses mecanismos de reparação heurística buscam manter as soluções vizinhas geradas durante as perturbações dentro do espaço de soluções viáveis do problema.

4.2.4 Exemplo Prático de Representação e Avaliação

Para ilustrar o funcionamento da representação binária e do avaliador incremental, considera-se uma instância simplificada, baseada na instância A20 do desafio proposto pelo Simpósio Brasileiro de Pesquisa Operacional em parceria com o Mercado Livre (Mercado Livre, 2025), composta por 5 pedidos (O_0 a O_4) e 5 corredores (A_0 a A_4). A Tabela 4.1 apresenta a estrutura dos dados dessa instância, especificando, na primeira coluna, a entidade analisada; na segunda coluna, a composição de cada pedido ou corredor indicada pelos itens e suas respectivas quantidades (I_k : quantidade); e, na terceira coluna, o total de itens contidos no pedido ou alocados no corredor. Ao final da tabela, constam os parâmetros de carga da coleta, com limite inferior $LB = 5$ e limite superior $UB = 12$.

Tabela 4.1 – Dados de entrada para exemplo de representação (baseado na instância A20).

Entidade	*Composição (Item: Quantidade)*	*Total de Itens*
Pedido O_0	$I_0 : 3, I_2 : 1$	4
Pedido O_1	$I_1 : 1, I_3 : 1$	2
Pedido O_2	$I_2 : 1, I_4 : 2$	3
Pedido O_3	$I_0 : 1, I_2 : 2, I_3 : 1, I_4 : 1$	5
Pedido O_4	$I_1 : 1$	1
Corredor A_0	$I_0 : 2, I_1 : 1, I_2 : 1, I_4 : 1$	5
Corredor A_1	$I_0 : 2, I_1 : 1, I_2 : 2, I_4 : 1$	6
Corredor A_2	$I_1 : 2, I_3 : 1, I_4 : 2$	5
Corredor A_3	$I_0 : 2, I_1 : 1, I_3 : 1, I_4 : 1$	5
Corredor A_4	$I_1 : 1, I_2 : 2, I_3 : 1, I_4 : 2$	6
Parâmetros de Carga: $LB = 5, UB = 12$		

A Figura 4.1 apresenta a representação computacional de uma solução candidata viável para essa instância por meio de dois vetores binários (*bitsets*). O vetor superior representa a seleção de pedidos (x_o), em que cada posição corresponde a um pedido específico (O_0 a O_4). O vetor inferior representa a ativação de corredores (y_a), indicando quais corredores físicos (A_0 a A_4) foram abertos.

Vetor de Pedidos (x_o)	0	1	1	0	0
Índices	O_0	O_1	O_2	O_3	O_4

Vetor de Corredores (y_a)	0	0	0	0	1
Índices	A_0	A_1	A_2	A_3	A_4

Figura 4.1 – Representação binária via *bitsets* para a solução do exemplo.

A partir da representação exibida na Figura 4.1, constata-se que os pedidos O_1 e O_2 estão selecionados ($x_o[1] = 1$ e $x_o[2] = 1$), totalizando uma carga de 5 itens, o que atende ao limite inferior $LB = 5$. O *StockBalanceEvaluator* verifica que a ativação exclusiva do corredor A_4 ($y_a[4] = 1$) provê estoque suficiente para suprir a demanda combinada desses pedidos. A produtividade dessa solução é calculada pela razão entre o total de itens selecionados (5) e o número de corredores ativos (1), resultando na função objetivo $Z = 5,0$, conforme definido pela Equação 3.1. Por outro lado, caso o algoritmo avaliasse a adição do pedido O_4 (composto por 1 unidade do item I_1), a carga total passaria para 6 itens. No entanto, como a única unidade do item I_1 disponível no corredor A_4 já foi consumida pelo pedido O_1, o *StockBalanceEvaluator* identifica que seria necessária a ativação de um corredor adicional para atender a essa nova demanda. Com 2 corredores ativos para suprir 6 itens, a produtividade da solução cairia para $Z = 3,0$ $(6/2)$. Esse comportamento ilustra claramente como o avaliador controla o saldo unitário de cada item e reflete o custo de abertura de novos corredores na função objetivo.

4.3 Heurísticas Construtivas

A etapa de construção da solução inicial exerce forte influência na qualidade da região do espaço de busca a ser explorada pelas metaheurísticas subsequentes. No *OOSP-MetaSolver Framework*, a geração de soluções iniciais viáveis opera sob duas perspectivas complementares de exploração do problema: centralizada nos pedidos (*Order-Centric*) e centralizada nos corredores (*Aisle-Centric*).

Para permitir uma investigação do compromisso entre velocidade computacional, intensificação gulosa e diversificação estrutural, cada uma dessas duas perspectivas foi estruturada para atuar sob três regimes algorítmicos distintos:

- **Estático (Guloso Determinístico a Priori):** O custo ou utilidade heurística dos elementos candidatos é calculado uma única vez antes do início da construção da *wave*. A seleção percorre estritamente essa ordem estática fixa, sem recálculos marginais à medida que o estoque ou os corredores abertos mudam.
- **Adaptativo (Guloso Dinâmico com Recálculo Marginal):** A cada novo elemento inserido na solução corrente, a função de avaliação recalcula dinamicamente o ganho marginal

dos candidatos remanescentes com base no estoque residual e nos corredores já ativados, selecionando de forma estritamente gulosa o elemento de maior utilidade momentânea.

- **Aleatorizado (Semi-Guloso via Lista Restrita de Candidatos):** Combina a avaliação dinâmica adaptativa com a introdução de aleatoriedade controlada por meio da Lista Restrita de Candidatos (*Restricted Candidate List*, RCL), modulada por um parâmetro de flexibilidade α.

Nas variantes aleatorizadas, a cada nova solução gerada, a metaheurística ou rotina controladora passa como parâmetro o valor de α a ser utilizado naquela construção. A estratégia de definição desse valor depende exclusivamente do método chamador, podendo variar ao longo das iterações de acordo com a política da metaheurística; caso nenhum valor seja explicitamente informado, o algoritmo construtivo adota o valor padrão de $\alpha = 0,10$. Na prática, durante a construção da solução, o parâmetro α recebido determina a porção dos elementos da Lista Restrita de Candidatos (RCL) elegível para seleção: após ordenar os candidatos com base em seus ganhos heurísticos marginais, o algoritmo sorteia o elemento que comporá a solução a partir dos elementos situados no topo da lista, cujo cardinal é delimitado por $\max(1, \lfloor \alpha \times |RCL| \rfloor)$. Dessa forma, construções operadas com valores menores de α aproximam o processo de uma escolha estritamente gulosa, enquanto valores maiores permitem a inclusão de candidatos de custo moderado, induzindo a diversificação no espaço de busca.

4.3.1 Construção Centrada no Pedido (*Order-Centric*)

A estratégia centralizada no pedido (*Order-Centric*) estrutura a construção da solução orientada pela produtividade individual de cada pedido do *backlog*. O objetivo consiste em priorizar a inserção dos pedidos que oferecem a maior relação entre quantidade de itens coletados e consumo de infraestrutura (corredores visitados). A exploração dessa estratégia processa-se em três regimes operacionais distintos:

4.3.1.1 Variação Estática Determinística (OC)

Na variação Estática Determinística (OC), a avaliação da utilidade dos pedidos ocorre uma única vez antes do início do processo construtivo. Cada pedido p recebe uma pontuação fixa calculada pela razão $\frac{I(p)}{A_{\min}(p)}$, onde $I(p)$ denota o número total de itens do pedido e $A_{\min}(p)$ representa o limite inferior de corredores necessários para atender a esse pedido, obtido mediante consulta prévia à estrutura de cache.

Algoritmo 4.1: Construção *Order-Centric* Estática Determinística (OC)

```
1  Solucao ← ∅, Estoque ← Disponibilidade Geral
2  ListaPedidos ← Ordenar backlog em ordem decrescente de I(p) / A_min(p)
3  for cada pedido ∈ ListaPedidos do
4      if I(Solucao) + I(pedido) ≤ UB ∧ Estoque satisfaz pedido then
5          Adicionar pedido à Solucao
6          Deduzir itens do pedido em Estoque e ativar corredores exigidos
7  return Solucao
```

O Algoritmo 4.1 apresenta complexidade temporal dominada pela ordenação inicial em $O(n \log n)$, onde n é o número de pedidos no *backlog*, seguida por uma etapa linear $O(n)$ de verificação e inserção. Por não realizar recálculos após a abertura de novos corredores, essa variante executa em tempo computacional reduzido, servindo como linha de base determinística.

4.3.1.2 Variação Adaptativa Gulosa (OC-A)

À medida que pedidos são inseridos na solução e novos corredores são ativados, pedidos pendentes cujos itens estejam alocados nos corredores já abertos passam a demandar menor custo adicional de infraestrutura. A variação Adaptativa Gulosa (OC-A) incorpora essa dinâmica ao recalcular marginalmente a produtividade dos pedidos remanescentes a cada passo construtivo. Para evitar o custo computacional quadrático $O(n^2)$ de reavaliar todo o *backlog* a cada inserção, o algoritmo emprega a técnica de atualização preguiçosa (*lazy update*) estruturada sobre uma Fila de Prioridades (***Binary Heap***).

Algoritmo 4.2: Construção *Order-Centric* Adaptativa Gulosa (OC-A)

```
1  Solucao ← ∅, Estoque ← Disponibilidade Geral
2  Estruturar FilaPrioridade de pedidos viáveis baseada no Limite Superior UB
3  while I(Solucao) < UB ∧ FilaPrioridade ≠ ∅ do
4      pedido ← Extrair topo de FilaPrioridade
5      if pontuação de pedido estiver desatualizada face aos corredores abertos then
6          ΔA ← Corredores adicionais necessários para atender pedido
7          if ΔA = 0 then
8              Φ(pedido) ← M + I(pedido)
9          else
10             Φ(pedido) ← I(pedido) / (ΔA)^2
11         Recolocar pedido com pontuação Φ(pedido) na FilaPrioridade e prosseguir
12     if estoque residual em Estoque satisfaz pedido then
13         Adicionar pedido à Solucao, alocar ΔA e atualizar Estoque
14 return Solucao
```

No Algoritmo 4.2, a verificação e atualização marginal ocorrem exclusivamente sobre o elemento posicionado no topo da fila no momento de sua extração (linhas 5 a 11). Se o pedido extraído possuir pontuação calculada sob um conjunto anterior de corredores abertos, sua produtividade $\Phi(pedido)$ é reavaliada em função de ΔA, que denota o número de corredores adicionais ainda fechados exigidos por aquele pedido.

Na formulação de pontuação, pedidos que podem ser atendidos integralmente pela infraestrutura já ativa ($\Delta A = 0$) recebem prioridade por meio da adição da constante superior M (definida no sistema como $M = 10^6$), favorecendo a preferência por inserções sem custo marginal de abertura de corredores (linha 8). Para pedidos que requerem a abertura de corredores adicionais ($\Delta A > 0$), aplica-se um decaimento quadrático em função de ΔA (linha 10). Essa penalização quadrática, em contraposição a um decaimento linear, foi formulada para penalizar a fragmentação de estoque: a razão inversamente proporcional a $(\Delta A)^2$ reduz a prioridade de pedidos que demandariam a abertura simultânea de múltiplos novos corredores, preservando a compacidade espacial da coleta.

4.3.1.3 Variação Aleatorizada com Lista Restrita de Candidatos (OC-R)

A variação Aleatorizada com Lista Restrita de Candidatos (OC-R) articula o recálculo marginal adaptativo com um mecanismo estocástico de seleção, com o intuito de diversificar a população de soluções iniciais.

Algoritmo 4.3: Construção *Order-Centric* Aleatorizada com Lista Restrita de Candidatos (OC-R)

Input: Fator de aleatoriedade α (recebido da metaheurística chamadora ou $\alpha = 0,10$ por padrão)

1 $Solucao \leftarrow \emptyset$, $Estoque \leftarrow$ Disponibilidade Geral
2 Estruturar $FilaPrioridade$ de pedidos viáveis baseada no Limite Superior UB
3 **while** $I(Solucao) < UB \wedge FilaPrioridade \neq \emptyset$ **do**
4 $CandidatosValidados \leftarrow \emptyset$
5 **while** $|CandidatosValidados| < k_{\max} \wedge FilaPrioridade \neq \emptyset$ **do**
6 Extrair $pedido$ do topo de $FilaPrioridade$ e aplicar atualização preguiçosa (*lazy update*)
7 **if** $pedido$ *possui pontuação atualizada e estoque viável em* $Estoque$ **then**
8 Adicionar $pedido$ a $CandidatosValidados$
9 **if** $CandidatosValidados = \emptyset$ **then**
10 **break**
11 Delimitar $RCL \subseteq CandidatosValidados$ aplicando restrição estocástica de fator α
12 Eleger aleatoriamente $pedido^* \in RCL$ sob distribuição uniforme e inseri-lo na $Solucao$
13 Devolver candidatos não selecionados à $FilaPrioridade$, alocar corredores e deduzir de $Estoque$
14 **return** $Solucao$

No Algoritmo 4.3, em cada iteração construtiva o algoritmo extrai e valida até $k_{\max}$ candidatos do topo da fila de prioridades, onde $k_{\max}$ limita o número de avaliações marginais por passo para preservar a eficiência computacional. Em seguida, delimita a Lista Restrita de Candidatos (RCL) aplicando o limiar α sobre as pontuações marginais atualizadas (linha 11) e seleciona estocasticamente um pedido $pedido^* \in RCL$ segundo distribuição uniforme (linha 12). Conforme determinado pela arquitetura do sistema, o parâmetro α é informado pelo método chamador em cada nova construção ou assume $\alpha = 0,10$ como valor padrão.

4.3.2 Construção Centrada no Corredor (*Aisle-Centric*)

Em instâncias caracterizadas por escassez de estoque ou dispersão de itens entre múltiplos corredores, a estratégia de construção centralizada no corredor (*Aisle-Centric*) direciona a busca para a infraestrutura de armazenagem. O objetivo consiste em selecionar corredores cujo estoque atenda ao maior volume possível da demanda pendente global do *backlog*, procedendo em seguida ao preenchimento dos pedidos viabilizados.

4.3.2.1 Variação Estática Determinística (AC)

Na variação Estática Determinística (AC), a utilidade inicial de cada corredor a é calculada uma única vez com base na intersecção entre o estoque do corredor e a demanda inicial global $D_0(i)$ de cada item i no *backlog*:

$$U_0(a) = \sum_{i \in a} \min(Estoque(a, i), D_0(i)) \tag{4.1}$$

Algoritmo 4.4: Construção *Aisle-Centric* Estática Determinística (AC)

```
1 Solucao ← ∅, CorredoresAbertos ← ∅
2 D_0 ← Somatório de itens de todos os pedidos do backlog
3 ListaCorredores ← Ordenar corredores decrescentemente pela utilidade inicial U_0(a)

4 for cada corredor ∈ ListaCorredores do
5     if I(Solucao) < UB then
6         Abrir corredor e incorporar seus itens ao EstoqueTotal
7         Executar sub-rotina de preenchimento guloso dos pedidos viabilizados na
            Solucao
8 return Solucao
```

Conforme demonstrado no Algoritmo 4.4, os corredores são abertos sequencialmente segundo a ordenação estática $U_0(a)$. Após a abertura de cada corredor, o algoritmo executa uma varredura gulosa sobre os pedidos ordenados por tamanho, inserindo na solução aqueles cuja demanda integral possa ser satisfeita pelo estoque agregado dos corredores ativos.

4.3.2.2 Variação Adaptativa Gulosa (AC-A)

Conforme corredores são abertos e pedidos são incorporados à solução, a demanda pendente global $D(i)$ se reduz. A variação Adaptativa Gulosa (AC-A) recalcula iterativamente a utilidade marginal dos corredores remanescentes exclusivamente em relação à demanda ainda não satisfeita, elegendo de forma gulosa o corredor de maior impacto marginal em cada passo.

Algoritmo 4.5: Construção *Aisle-Centric* Adaptativa Gulosa (AC-A)

```
1  Solucao ← ∅, CorredoresAbertos ← ∅
2  D ← Somatório de itens de todos os pedidos do backlog
3  while I(Solucao) < UB ∧ CorredoresDisponiveis ≠ ∅ do
4      forall corredor ∈ CorredoresDisponiveis do
5          U(corredor) ← Σ_i min(Estoque(corredor, i), D(i))
6      Eleger de forma estritamente gulosa o corredor* de maior U(corredor) e abrir na
       Solucao
7      Atualizar EstoqueTotal com os itens providos por corredor*
8      forall pedido ∈ CandidatosOrdenadosPorTamanho do
9          if pedido ∉ Solucao ∧ I(Solucao ∪ {pedido}) ≤ UB then
10             if EstoqueTotal satisfaz integralmente a demanda de pedido then
11                 Adicionar pedido à Solucao e deduzir de EstoqueTotal
12                 Deduzir itens do pedido do vetor de demanda pendente D
13 return Solucao
```

O Algoritmo 4.5 inicia com as estruturas vazias (linha 1). Durante o laço principal (linha 3), a utilidade marginal de cada corredor disponível é avaliada pelo seu estoque útil perante a demanda pendente residual $D(i)$ (linha 5). Após a ativação do corredor que apresenta a maior pontuação, executa-se o preenchimento guloso em cascata (linhas 8 a 12), deduzindo os itens dos pedidos atendidos tanto do estoque agregado quanto do vetor de demanda pendente D.

4.3.2.3 Variação Aleatorizada com Lista Restrita de Candidatos (AC-R)

A variação Aleatorizada com Lista Restrita de Candidatos (AC-R) incorpora a restrição estocástica de fator α na escolha do próximo corredor a ser aberto, diversificando os subconjuntos de corredores ativados nas fases iniciais.

Algoritmo 4.6: Construção *Aisle-Centric* Aleatorizada com Lista Restrita de Candidatos (AC-R)

Input: Fator de aleatoriedade α (recebido da metaheurística chamadora ou $\alpha = 0,10$ por padrão)

1 $Solucao \leftarrow \emptyset$, $CorredoresAbertos \leftarrow \emptyset$
2 $D \leftarrow$ Somatório de itens de todos os pedidos do backlog
3 **while** $I(Solucao) < UB \wedge CorredoresDisponiveis \neq \emptyset$ **do**
4 **forall** $corredor \in CorredoresDisponiveis$ **do**
5 $U(corredor) \leftarrow \sum_i \min(Estoque(corredor, i), D(i))$
6 Delimitar RCL aplicando restrição estocástica de fator α aos corredores
7 Eleger aleatoriamente $corredor^* \in RCL$ sob distribuição uniforme e abrir na $Solucao$
8 Atualizar $EstoqueTotal$ com os itens providos por $corredor^*$
9 Executar preenchimento guloso dos pedidos viabilizados na $Solucao$ e abater de D
10 **return** $Solucao$

No Algoritmo 4.6, após o cálculo das utilidades marginais dos corredores perante a demanda residual D, constitui-se a RCL com os corredores delimitados pelo limiar α (linha 6). A escolha estocástica de $corredor^* \in RCL$ sob distribuição uniforme (linha 7) permite explorar configurações alternativas de corredores antes de proceder à sub-rotina de preenchimento guloso de pedidos.

4.3.3 Hibridização Estocástica entre Estratégias Construtivas

Embora as heurísticas *Order-Centric* e *Aisle-Centric* possuam lógicas de construção diferentes e complementares, o *framework* também incorpora um mecanismo de aleatoriedade guiada no nível estratégico. Ao iniciar a construção de uma solução (ou população inicial), o algoritmo utiliza uma probabilidade predefinida para intercalar dinamicamente entre essas duas estratégias. Essa alternância estocástica enriquece a diversidade das soluções geradas inicialmente, fornecendo às metaheurísticas subsequentes um conjunto mais amplo e variado de pontos de partida para o processo de busca.

4.3.4 Formalização das Variações Construtivas Avaliadas

Para fins de investigação experimental e isolamento do impacto do parâmetro de aleatoriedade α, as variantes construtivas *Order-Centric* e *Aisle-Centric* são instanciadas em três variações operacionais distintas:

- **Variação Estática ou Determinística (OC e AC):** Avalia e ordena os candidatos, sejam pedidos ou corredores, uma única vez no início da execução. Em seguida, percorre sequen-

cialmente essa lista fixa até o preenchimento da *wave*, sem recalcular pontuações após atualizações de estoque ou abertura de novos corredores.

- **Variação Gulosa Adaptativa (OC-A e AC-A):** A cada passo construtivo, recalcula dinamicamente a utilidade marginal dos candidatos restantes com base no estado atual da solução. No caso da variação adaptativa centrada em corredores, o algoritmo abate os itens já consolidados na coleta; no caso da variação adaptativa centrada em pedidos, atualiza o custo marginal decorrente de eventuais novos corredores necessários. Em seguida, seleciona de forma estritamente gulosa o candidato de maior pontuação naquela iteração. Nas análises comparativas deste trabalho, a variação *Order-Centric* adaptativa é denotada sistematicamente por OC-A, enquanto a variação *Aisle-Centric* adaptativa é denotada por AC-A.

- **Variação Aleatorizada com Lista Restrita de Candidatos (OC-R e AC-R):** Incorpora aleatoriedade controlada e Lista Restrita de Candidatos modulada por um parâmetro α na escolha dos elementos. O valor de α permanece fixo durante a construção de uma solução individual, sendo variado entre as iterações pela metaheurística controladora para diversificar as rotas geradas e evitar ótimos locais estritamente gulosos.

4.4 Buscas Locais e Metaheurísticas Clássicas

A busca local no *OOSP-MetaSolver Framework* atua por meio de modificações diretas na representação vetorial das soluções. Essas modificações alteram sistematicamente a solução corrente e acionam rotinas de reparo para manter a factibilidade operacional, com o objetivo de identificar vizinhos que aprimorem o valor da função objetivo.

A exploração do espaço de soluções no framework é conduzida por quatro estruturas de vizinhança principais, que podem atuar de forma especializada em um único domínio ou ser combinadas de forma multidimensional, como ocorre no *Variable Neighborhood Descent* (VND), alternando estrategicamente entre a manipulação de pedidos (dimensão de itens) e de corredores (dimensão de custo):

1. **Inserção de Pedido (*Order Insertion*):** Avalia a inclusão de um pedido pendente do *backlog* na solução corrente. Caso a inserção resulte em déficit de estoque, a heurística de reparo do avaliador incremental ativa os corredores necessários para restaurar a viabilidade.

2. **Permuta de Pedidos (*Order Swap*):** Realiza a substituição de um pedido atualmente selecionado por outro do *backlog*, permitindo trocar conjuntos de itens por opções com maior produtividade marginal.

3. **Remoção de Corredor (*Aisle Removal*):** Avalia a desativação de um corredor ativo para reduzir o denominador da função objetivo. Caso a remoção torne a solução inviável, o

sistema remove os pedidos dependentes daquele estoque para restabelecer a integridade da coleta.

4. **Permuta de Corredores (*Aisle Swap*):** Substitui um corredor ativo por outro que ofereça maior densidade de itens demandados, sincronizando os pedidos associados para manter a viabilidade.

Para guiar a exploração, adota-se a estratégia de primeira melhoria (*First Improvement*). Em vez de inspecionar exaustivamente toda a vizinhança em busca da transição de maior ganho (*Best Improvement*), o algoritmo aceita e aplica imediatamente o primeiro movimento viável que resulte em melhoria estrita da função objetivo, proporcionando transições mais ágeis no espaço de busca.

Em instâncias de grande porte, a avaliação exaustiva de todas as permutações viáveis apresenta custo computacional proibitivo. Por essa razão, o *framework* emprega uma estratégia de amostragem aleatória da vizinhança (*Randomized Neighborhood Sampling*). Em cada iteração da busca local, o avaliador inspeciona um subconjunto estocástico fixado em até 5.000 movimentos candidatos. Esse tamanho de amostra foi definido por meio de testes exploratórios durante o desenvolvimento, buscando um compromisso entre a probabilidade de identificar vizinhos de melhoria e o custo computacional por iteração, o que viabiliza o tempo de resposta nas instâncias de maior dimensão.

Algoritmo 4.7: Mecanismo de Exploração de Vizinhança com Sincronização e Reparo

Input: Solução corrente $s = (x_o, y_a)$, estrutura de vizinhança N, tamanho da amostra $k_{\max} = 5000$

1 $s^* \leftarrow s$
2 $Candidatos \leftarrow$ AmostraAleatoria$(s, N, k_{\max})$
3 **forall** $movimento \in Candidatos$ **do**
4 Registrar ponto de controle no histórico de reversão de saldos
5 $s' \leftarrow s$
6 **if** $N =$ *Inserção de Pedido* **then**
7 Seja p o pedido candidato de $movimento$
8 **if** $x'_o[p] = 0$ **then**
9 $x'_o[p] \leftarrow 1$
10 SincronizarCorredoresParaPedido(s', p)
11 **else if** $N =$ *Permuta de Pedidos* **then**
12 Seja $(p_{\text{sai}}, p_{\text{entra}})$ o par de pedidos de $movimento$
13 $x'_o[p_{\text{sai}}] \leftarrow 0$; $x'_o[p_{\text{entra}}] \leftarrow 1$
14 SincronizarCorredoresParaPedido(s', p_{entra})
15 **else if** $N =$ *Remoção de Corredor* **then**
16 Seja a o corredor de $movimento$
17 $y'_a[a] \leftarrow 0$
18 SincronizarPedidosAposRemocaoCorredor(s', a)
19 **else if** $N =$ *Permuta de Corredores* **then**
20 Seja $(a_{\text{sai}}, a_{\text{entra}})$ o par de corredores de $movimento$
21 $y'_a[a_{\text{sai}}] \leftarrow 0$; $y'_a[a_{\text{entra}}] \leftarrow 1$
22 SincronizarPedidosAposRemocaoCorredor(s', a_{sai})
23 **if** *Viavel*$(s') \wedge f(s') > f(s^*)$ **then**
24 $s^* \leftarrow s'$
25 **return** s^* `// Estratégia First Improvement`
26 Reverter alterações de saldos até o ponto de controle registrado `// Tempo O(k)`
27 **return** s^*

O Algoritmo 4.7 formaliza o procedimento de exploração de vizinhança sob amostragem aleatória e reparo automático. A partir da solução corrente s (linha 1), o algoritmo extrai um subconjunto estocástico de até $k_{\max} = 5000$ movimentos candidatos para a estrutura N (linha 2). Esse teto amostral foi definido empiricamente para limitar o tempo de processamento em instâncias de grande porte, evitando o custo computacional quadrático da exploração exaustiva. Antes de alterar a representação binária, registra-se um ponto de controle no histórico incremental do avaliador (linha 4). Conforme a estrutura de vizinhança investigada, Inserção de Pedido (linha 6), Permuta de Pedidos (linha 11), Remoção de Corredor (linha 15) ou Permuta

de Corredores (linha 19), as rotinas de sincronização ajustam a infraestrutura ativa ou removem pedidos inviáveis para preservar a factibilidade. Ao identificar a primeira transição viável s' cuja produtividade $f(s')$ supere estritamente $f(s^*)$ (linha 23), adota-se a estratégia de primeira melhoria (*First Improvement*), retornando a nova solução (linha 25). Caso o movimento não resulte em aprimoramento, o histórico é revertido em tempo proporcional ao número estrito k de itens modificados (linha 26).

4.4.1 Variable Neighborhood Descent

Como as buscas locais baseadas em uma única estrutura de vizinhança podem estagnar prematuramente em ótimos locais, o *framework* incorpora em seu acervo modular o algoritmo de Descida em Vizinhança Variável (*Variable Neighborhood Descent*, VND) para possibilitar a exploração de múltiplas vizinhanças.

Aplicado como uma etapa de refinamento, o VND atua de forma integrada sobre as dimensões de pedidos e corredores. A estratégia prioriza inicialmente movimentos que alteram os itens da solução, realizando Inserções e, em seguida, Permutas de Pedidos. Caso essas estruturas não consigam produzir melhorias, indicando que não há como mais melhorar os pedidos sob a solução atual, a busca avança para a dimensão espacial. Nesse ponto, o algoritmo tenta remover corredores ociosos ou com baixo aproveitamento de estoque (Remoção de Corredores) e, se necessário, realiza a troca de corredores (Permuta de Corredores) para encontrar uma configuração de infraestrutura mais enxuta. Sempre que qualquer uma dessas quatro vizinhanças encontra uma solução estritamente melhor, o ciclo reinicia a partir da primeira vizinhança, promovendo uma exploração contínua até alcançar um ótimo local em relação a todas as estruturas simultaneamente.

Algoritmo 4.8: Busca Local *Variable Neighborhood Descent* (VND)

Input: Solução inicial s

1 $k \leftarrow 1$
2 Definir conjunto ordenado de vizinhanças $N \leftarrow (N_1, N_2, N_3, N_4)$, onde:
3 $N_1 =$ Inserção de Pedido, $N_2 =$ Permuta de Pedidos, $N_3 =$ Remoção de Corredor, $N_4 =$ Permuta de Corredores
4 **while** $k \leq |N| \wedge$ *TempoLimite não excedido* **do**
5 $s' \leftarrow$ ExplorarVizinhancaPrimeiraMelhoria(s, N_k)
6 **if** $f(s') > f(s)$ **then**
7 $s \leftarrow s'$
8 $k \leftarrow 1$ `// Retorna à primeira vizinhança após melhoria`
9 **else**
10 $k \leftarrow k + 1$ `// Avança para a próxima vizinhança`
11 **return** s

O Algoritmo 4.8 detalha o funcionamento do VND no refinamento de soluções. A rotina

inicializa o índice $k = 1$ (linha 1) e estrutura a sequência ordenada de vizinhanças N (linha 2). Enquanto as estruturas não se esgotarem ($k \leq |N|$) e o tempo limite de execução não for atingido (linha 4), o algoritmo explora a estrutura N_k sob a estratégia de primeira melhoria (linha 5). Caso seja encontrada uma solução s' com produtividade estritamente superior ($f(s') > f(s)$ na linha 6), a solução corrente é atualizada (linha 7) e a exploração reinicia a partir da primeira vizinhança ($k = 1$ na linha 8). Na ausência de aprimoramento, o algoritmo avança para a estrutura subsequente ($k \leftarrow k + 1$ na linha 10).

4.4.2 Busca Tabu

A metaheurística Busca Tabu é incorporada ao conjunto de estratégias de refinamento local como um método de trajetória única dotado de memória explícita de curto prazo (Glover; Laguna, 1997). Diferentemente da descida pura, a Busca Tabu aceita transições que temporariamente reduzam o valor da função objetivo para escapar de ótimos locais.

Para evitar a ciclagem em regiões recém-visitadas, o algoritmo mantém uma Lista Tabu (T) que registra os atributos das transformações recentes, especificamente os identificadores dos pedidos ou corredores alterados. Um atributo permanece classificado como proibido por um número de iterações regulado pelo parâmetro de expiração tabu (τ). No contexto das instâncias avaliadas, adota-se uma heurística de escala em função do total de itens da instância ($|I|$), formulada como $\tau = \min(15, \lfloor\sqrt{|I|}\rfloor)$. Essa regra calibra a memória restritiva proporcionalmente à raiz quadrada da dimensão do problema, aplicando um teto superior de 15 iterações para evitar o bloqueio excessivo de movimentos em instâncias de grande porte. Para prevenir a rejeição de soluções promissoras, aplica-se o critério de aspiração: um movimento proibido tem seu status tabu ignorado caso conduza a uma solução estritamente superior à melhor solução global conhecida (s^*). O refinamento é conduzido por até $iter_{\max} = 500$ iterações, limiar definido por testes exploratórios durante o desenvolvimento para equilibrar refinamento e tempo de execução, sendo interrompido antecipadamente em caso de esgotamento do tempo limite global.

Algoritmo 4.9: Busca Tabu para Refinamento de Soluções do PSPO

Input: Solução inicial s, expiração tabu $\tau = \min(15, \lfloor\sqrt{|I|}\rfloor)$, limite de iterações $iter_{\max} = 500$

1 $s^* \leftarrow s$
2 $T \leftarrow \emptyset$, $iter \leftarrow 0$ `// T representa o conjunto de atributos proibidos`
3 **while** $iter < iter_{\max} \wedge$ *TempoLimite não excedido* **do**
4 $MelhorVizinho \leftarrow \emptyset$, $MelhorScoreVizinho \leftarrow -\infty$
5 $MovimentoEleito \leftarrow$ Nulo
6 $Amostra \leftarrow$ AmostraAleatoria$(s, N, 5000)$
7 **forall** $mov \in Amostra$ **do**
8 $s' \leftarrow$ AplicarMovimento(s, mov)
9 $EhTabu \leftarrow (mov.$atributo $\in T)$
10 $Aspirado \leftarrow (f(s') > f(s^*))$
11 **if** $(\neg EhTabu \vee Aspirado) \wedge$ *Viavel*(s') **then**
12 **if** $f(s') > MelhorScoreVizinho$ **then**
13 $MelhorVizinho \leftarrow s'$, $MelhorScoreVizinho \leftarrow f(s')$
14 $MovimentoEleito \leftarrow mov$
15 **if** $MelhorVizinho = \emptyset$ **then**
16 **break** `// Nenhum movimento viável e não proibido encontrado`
17 $s \leftarrow MelhorVizinho$
18 **if** $f(s) > f(s^*)$ **then**
19 $s^* \leftarrow s$
20 $T \leftarrow T \cup \{(MovimentoEleito.$atributo_inverso$, iter + \tau)\}$
21 $T \leftarrow \{(a, exp) \in T \mid exp > iter\}$ `// Remove atributos cuja critério expirou`
22 $iter \leftarrow iter + 1$
23 **return** s^*

O Algoritmo 4.9 sumariza o fluxo operacional da Busca Tabu no refinamento de rotas. Inicializada a partir de s (linha 1), a rotina inspeciona em cada passo iterativo (linha 3) uma amostra aleatória restrita ao limite de 5.000 movimentos candidatos (linha 6), teto adotado para controlar a explosão combinatória em espaços de busca com milhares de itens. O algoritmo aplica cada modificação (linha 8) e seleciona o vizinho viável que apresente o maior valor na função objetivo (linha 12), desde que seu atributo não conste em T ou satisfaça o critério de aspiração ao superar a melhor solução conhecida s^* (linha 10). Ao efetuar a transição eleita, o atributo inverso é inserido na lista T com validade de τ passos adicionais (linha 20), impedindo reversões imediatas ao longo das $iter_{\max}$ iterações permitidas.

4.4.3 Late Acceptance Hill Climbing (LAHC)

A metaheurística *Late Acceptance Hill Climbing* (LAHC) é uma metaheurística de trajetória única proposta por Burke e Bykov (2017) que transpõe platôs e ótimos locais sem a necessidade de esquemas de resfriamento ou parâmetros probabilísticos.

O mecanismo fundamental do LAHC baseia-se na comparação da função objetivo da solução candidata s' com a solução visitada há L iterações no passado. Para tanto, o algoritmo gerencia um vetor circular de histórico de aptidões F_{hist} de comprimento fixo $L = 50$. Esse valor adota o padrão consagrado na literatura da metaheurística (Burke; Bykov, 2017), corroborado em testes exploratórios durante o desenvolvimento por proporcionar equilíbrio entre capacidade de escape de ótimos locais e velocidade de convergência. Um movimento vizinho s' é aceito se sua aptidão for maior ou igual ao valor registrado no histórico na posição $v = iter \pmod{L}$, de modo que $f(s') \geq F_{hist}[v]$. O processo iterativo de exploração é executado por um limite nominal de $iter_{\max} = 1.000$ iterações, valor fixado por testes empíricos para favorecer o refinamento local sem violar o tempo limite de execução.

Complementarmente às lógicas de aceitação, métodos de trajetória única estocásticos como o LAHC demandam uma função geradora de transições. O Algoritmo 4.10 formaliza a rotina de perturbação utilizada para amostrar um vizinho válido durante a busca local. Em vez de avaliar todo o conjunto vizinho, o algoritmo sorteia estocasticamente um único movimento inerente à estrutura topológica ativa N. O procedimento impõe um teto empírico de $LimiteTentativas = 50$ iterações restritivas (linhas 3-8), calibrado mediante testes exploratórios para evitar aprisionamento computacional (ciclagem infinita) em espaços de factibilidade muito estreitos. A cada passo, aplica-se o movimento sorteado (linha 5) e sua factibilidade dimensional é testada; a primeira modificação viável é prontamente retornada à metaheurística (linha 7). Caso o teto de 50 tentativas se esgote sem a formulação de um vizinho válido, a rotina estabiliza a transição retornando a própria solução inicial s (linha 9), assegurando a continuidade do processo sem violações.

Algoritmo 4.10: Geração de Vizinho Aleatório Viável

Input: Solução corrente s, estrutura de vizinhança N

$Tentativas \leftarrow 0$
$LimiteTentativas \leftarrow 50$ `// Teto de segurança empírico`
while $Tentativas < LimiteTentativas$ **do**
  $mov \leftarrow$ AmostraMovimento(s, N) `// Sorteia um movimento conforme a estrutura`
  $s' \leftarrow$ AplicarMovimento(s, mov)
  if *Viavel*(s') **then**
    return s'
  $Tentativas \leftarrow Tentativas + 1$
return s `// Retorna a solução original em caso de falha de factibilidade`

Algoritmo 4.11: *Late Acceptance Hill Climbing* (LAHC)

Input: Solução inicial s, comprimento do histórico $L = 50$, limite de iterações $iter_{\max} = 1000$

1 $s^* \leftarrow s$
2 **forall** $i \in \{0, \ldots, L-1\}$ **do**
3 $F_{hist}[i] \leftarrow f(s)$ `// Inicializa o histórico com a produtividade inicial`
4 $iter \leftarrow 0$
5 **while** $iter < iter_{\max} \wedge$ *TempoLimite não excedido* **do**
6 $s' \leftarrow$ GerarVizinhoAleatorio(s, N) `// Perturbação viável amostrada da vizinhança`
7 $v \leftarrow iter \pmod{L}$
8 **if** $f(s') \geq F_{hist}[v] \vee f(s') \geq f(s)$ **then**
9 $s \leftarrow s'$
10 **if** $f(s) > f(s^*)$ **then**
11 $s^* \leftarrow s$
12 $F_{hist}[v] \leftarrow f(s)$ `// Atualiza a memória de aceitação atrasada`
13 $iter \leftarrow iter + 1$
14 **return** s^*

O Algoritmo 4.11 detalha a execução do LAHC. A partir da solução corrente s (linha 1), o vetor circular de histórico F_{hist} é inicializado na linha 2 com o valor da função objetivo atual. Em cada passo do laço principal (linha 5), gera-se uma perturbação viável aleatória s' (linha 6). A nova solução substitui a corrente na linha 8 caso apresente produtividade superior ou igual ao valor armazenado na posição de atraso L ($F_{hist}[v]$) ou caso supere a própria solução corrente.

Ao término do passo, a respectiva célula do histórico é atualizada na linha 12 com a aptidão da solução aceita, permitindo transpor regiões de estagnação ao longo das $iter_{\max}$ iterações.

4.5 Metaheurísticas Reativas

O *OOSP-MetaSolver Framework* combina duas estratégias de busca global: uma metaheurística baseada em trajetórias (GRASP Reativo) e outra baseada em populações (*Adaptive Reactive Binary Particle Swarm Optimization* — AR-BPSO), que exploram o espaço de soluções de forma complementar.

4.5.1 GRASP Reativo (R-GRASP)

O procedimento GRASP Reativo (R-GRASP) incorpora um mecanismo de aprendizado adaptativo para ajustar os parâmetros de construção com base no desempenho observado durante a busca.

O R-GRASP alterna dinamicamente entre as abordagens construtivas *Order-Centric* e *Aisle-Centric* conforme a qualidade das soluções geradas. Inicialmente, o algoritmo estabelece probabilidades de escolha equilibradas ($p_{OC} = 0,5$ e $p_{AC} = 0,5$) para ambas as heurísticas. Em cada iteração construtiva, o parâmetro de aleatoriedade α é amostrado uniformemente no intervalo $[\ 0,05\ ,\ 0,30\ ]$, faixa adotada na literatura do GRASP e corroborada em testes exploratórios por proporcionar equilíbrio entre qualidade gulosa e diversificação das soluções iniciais.

A seleção entre as heurísticas construtivas é regida por probabilidades dinâmicas atualizadas periodicamente em blocos de 20 iterações (*chunks*), intervalo definido por testes exploratórios durante o desenvolvimento para acumular uma amostra estatisticamente representativa antes da reponderação. Ao término de cada bloco, avalia-se o desempenho médio das soluções geradas por cada método em comparação com a melhor solução encontrada até o momento. A produtividade média de cada estratégia no bloco é normalizada em relação ao melhor resultado global e elevada ao expoente de reatividade $q = 6$. Esse valor foi calibrado por testes exploratórios durante o desenvolvimento, buscando um compromisso entre amplificar as diferenças de produtividade média das estratégias e evitar a exclusão prematura de qualquer variante construtiva.

Para suavizar oscilações decorrentes de iterações atípicas, as probabilidades são atualizadas por meio de uma Média Móvel Exponencial (EMA) com taxa de aprendizado de 0,20, definida empiricamente para manter a estabilidade sem comprometer a responsividade da busca. Esse mecanismo permite priorizar a técnica de construção mais eficaz para as características da instância em processamento. O R-GRASP opera com um limite nominal de 2.000 iterações totais, teto estabelecido para viabilizar a convergência em instâncias de maior porte, sendo a execução interrompida antecipadamente caso atinja o tempo limite de 600 segundos.

Algoritmo 4.12: GRASP Reativo com Média Móvel Exponencial (R-GRASP)

1 $s^* \leftarrow \emptyset$, $f(s^*) \leftarrow -\infty$
2 $p_{OC} \leftarrow 0,5$, $p_{AC} \leftarrow 0,5$ `// Probabilidades iniciais equilibradas`
3 $SomaAptidao_{OC} \leftarrow 0$, $Cont_{OC} \leftarrow 0$; $SomaAptidao_{AC} \leftarrow 0$, $Cont_{AC} \leftarrow 0$
4 $iter \leftarrow 1$
5 **while** $iter \leq 2000 \wedge$ *TempoLimite não excedido* **do**
6 Sorteio $\gamma \sim U(0,1)$
7 **if** $\gamma \leq p_{OC}$ **then**
8 $s \leftarrow$ ConstrucaoAleatorizadaOC($\alpha \sim U(0,05; 0,30)$)
9 $EstrategiaUsada \leftarrow$ OC
10 **else**
11 $s \leftarrow$ ConstrucaoAleatorizadaAC($\alpha \sim U(0,05; 0,30)$)
12 $EstrategiaUsada \leftarrow$ AC
13 $s \leftarrow$ BuscaLocalVND(s) `// Refinamento intensificador multi-vizinhança`
14 **if** $EstrategiaUsada =$ *OC* **then**
15 $SomaAptidao_{OC} \leftarrow SomaAptidao_{OC} + f(s)$; $Cont_{OC} \leftarrow Cont_{OC} + 1$
16 **else**
17 $SomaAptidao_{AC} \leftarrow SomaAptidao_{AC} + f(s)$; $Cont_{AC} \leftarrow Cont_{AC} + 1$
18 **if** $f(s) > f(s^*)$ **then**
19 $s^* \leftarrow s$; $f(s^*) \leftarrow f(s)$
20 `// Atualização Reativa periódica das probabilidades em blocos de 20 iterações`
21 **if** $iter \pmod{20} = 0$ **then**
22 $Media_{OC} \leftarrow \frac{SomaAptidao_{OC}}{\max(1, Cont_{OC})}$; $Media_{AC} \leftarrow \frac{SomaAptidao_{AC}}{\max(1, Cont_{AC})}$
23 $q \leftarrow 6$ `// Expoente de amplificação de reatividade`
24 $Apt_{OC} \leftarrow \left(\frac{Media_{OC}}{f(s^*)}\right)^q$; $Apt_{AC} \leftarrow \left(\frac{Media_{AC}}{f(s^*)}\right)^q$
25 $p_{OC}^{novo} \leftarrow \frac{Apt_{OC}}{Apt_{OC}+Apt_{AC}}$; $p_{AC}^{novo} \leftarrow 1 - p_{OC}^{novo}$
26 $p_{OC} \leftarrow 0,80 \cdot p_{OC} + 0,20 \cdot p_{OC}^{novo}$ `// Suavização via EMA`
27 $p_{AC} \leftarrow 1 - p_{OC}$
28 $SomaAptidao_{OC} \leftarrow 0, Cont_{OC} \leftarrow 0, SomaAptidao_{AC} \leftarrow 0, Cont_{AC} \leftarrow 0$
29 $iter \leftarrow iter + 1$
30 **return** s^*

O Algoritmo 4.12 formaliza a execução do R-GRASP. Inicializado na linha 1, o laço principal (linha 5) seleciona probabilisticamente entre a construção orientada a pedido ou a corredor com base em p_{OC} e p_{AC} (linha 6). Após a geração da solução inicial, aplica-se o

refinamento multi-vizinhança pelo algoritmo VND na linha 13, e as aptidões resultantes são acumuladas nos registros estatísticos na linha 14. Caso a solução aprimore o incumbente global, atualiza-se s^* na linha 18. Ao completar cada bloco de 20 iterações (linha 21), o sistema avalia a produtividade média recente de cada variante, eleva a razão de desempenho ao expoente de amplificação $q = 6$ e atualiza as probabilidades de seleção por meio de Média Móvel Exponencial com fator de suavização de 0,20. O ciclo prossegue até o limite nominal de 2.000 iterações ou o atingimento do tempo computacional máximo.

4.5.2 Adaptive Reactive Binary Particle Swarm Optimization (AR-BPSO)

Como abordagem populacional complementar, o *framework* incorpora a Otimização por Enxame de Partículas (*Particle Swarm Optimization* — PSO), adaptada para espaços de busca discretos por meio de sua formulação binária (BPSO) (Kennedy; Eberhart, 1997).

A evolução do enxame ocorre em um espaço multidimensional discreto, onde cada partícula representa uma solução candidata. No AR-BPSO, a população inicial de soluções é construída a partir de heurísticas aleatorizadas e, simultaneamente, cada partícula é alocada estocasticamente para focar sua busca iterativa na dimensão de pedidos ou na dimensão de corredores, assumindo uma probabilidade inicial equilibrada de 0,5. Dessa forma, o enxame explora ambas as representações do problema em paralelo. O movimento das partículas é regido por um vetor de velocidade, que influencia a transição de estado de cada variável binária. Cada partícula mantém em memória sua melhor posição individual (*pbest*), enquanto o enxame compartilha a melhor posição global identificada (*gbest*). Esse processo de compartilhamento de informações direciona a convergência da população para regiões promissoras do espaço de busca.

A dinâmica de atualização de uma partícula i para o elemento j de sua dimensão ativa de busca, seja ele o índice de um pedido ou de um corredor, é definida pela sua velocidade v_{ij}^{t+1}, conforme a Equação 4.2:

$$v_{ij}^{t+1} = w v_{ij}^{t} + c_1 r_1 (pbest_{ij} - x_{ij}^{t}) + c_2 r_2 (gbest_j - x_{ij}^{t}) \tag{4.2}$$

Nessa formulação, o fator de inércia w pondera a influência da velocidade anterior v_{ij}^{t}, e os coeficientes c_1 e c_2 regulam o equilíbrio entre a exploração individual e a convergência social, enquanto r_1 e r_2 representam variáveis aleatórias amostradas uniformemente no intervalo $[0, 1]$ a cada iteração, conferindo o caráter estocástico à busca. As velocidades são mapeadas para o espaço binário por meio de uma função sigmoide e limitadas ao intervalo $[-4,0; 4,0]$, de modo que $v_{\max} = 4,0$. Esse limite previne a saturação da sigmoide em probabilidades extremas como 0 ou 1, preservando a capacidade de mutação dos bits ao longo das gerações.

Para adaptar as velocidades contínuas à busca discreta sem gerar soluções inviáveis, o AR-BPSO adota uma formulação guiada por prioridades. Em abordagens binárias tradicionais, a função sigmoide $\text{sig}(v_{ij}) = \frac{1}{1+e^{-v_{ij}}}$ define a probabilidade independente de ativação de cada

bit, o que frequentemente violaria as restrições de capacidade ou estoque da *wave*. No AR-BPSO, os valores transformados pela sigmoide são interpretados como escores de prioridade ($Escore_j$). Em cada iteração, os elementos da dimensão correspondente (pedidos ou corredores) são ordenados de acordo com seus escores $Escore_j$, e uma sub-rotina construtiva guiada insere sequencialmente os elementos prioritários na solução, respeitando integralmente as restrições de estoque e os limites de carga (LB e UB). A alocação de uma partícula para a dimensão de pedidos ou de corredores define qual vetor binário constitui o alvo primário da perturbação de velocidades em cada iteração; o vetor complementar é ajustado passivamente pelas heurísticas de reparo e sincronização para manter a viabilidade da solução. Além disso, o algoritmo monitora continuamente a taxa de estagnação da melhor solução global em relação ao limite de gerações, transitando reativamente entre três estados comportamentais distintos:

1. **Estado de Convergência:** Ativado quando a taxa de estagnação da melhor solução global é inferior a 30% das gerações máximas. Mantém os coeficientes cognitivo e social equilibrados, com $c_1 = 1,494$ e $c_2 = 1,494$, conforme a calibração padrão de convergência proposta na literatura para otimização por enxame de partículas. Durante essa fase operacional, a probabilidade de aplicar o refinamento por busca local em cada partícula é fixada em seu valor base de 15%, calibrado empiricamente para balancear o refinamento das soluções candidatas e o tempo de execução.

2. **Estado de Diversificação:** Acionado quando a estagnação situa-se entre 30% e 60% das gerações máximas, esses limiares foram determinados por meio de testes exploratórios durante o desenvolvimento. Prioriza a componente cognitiva individual, ajustando os coeficientes para $c_1 = 2,0$ e $c_2 = 1,0$, com o objetivo de incentivar a exploração de novas áreas do espaço de busca pelas partículas.

3. **Estado de Alta Turbulência:** Deflagrado quando a estagnação excede 60% das gerações máximas. Nesse estado, os coeficientes mantêm-se ajustados em $c_1 = 2,0$ e $c_2 = 1,0$, mas o algoritmo introduz perturbações estocásticas no vetor de velocidades e eleva a probabilidade de aplicação da busca local de 15% para 30%, parâmetros calibrados empiricamente visando superar a estagnação por meio de reconfigurações estruturais.

Algoritmo 4.13: *Adaptive Reactive Binary Particle Swarm Optimization* (AR-BPSO)

1 Inicializar nuvem de partículas aleatoriamente utilizando métodos construtivos
2 **while** *Tempo limite não excedido e Critério de Parada não atingido* **do**
3 Analisar Taxa de Estagnação da melhor solução global ($gbest$)
4 Ajustar estado: Convergência, Diversificação ou Alta Turbulência (ajuste de c_1, c_2, w, turbulência)
5 **forall** *partícula i na nuvem (em paralelo)* **do**
6 **forall** *dimensão j* **do**
7 Calcular velocidade v_{ij}^{t+1} conforme a Equação 4.2
8 $v_{ij}^{t+1} \leftarrow \max(-v_{\max}, \min(v_{\max}, v_{ij}^{t+1}))$
9 Atribuir escore sigmoide $Escore_j \leftarrow \frac{1}{1+e^{-v_{ij}^{t+1}}}$
10 Aplicar perturbação estocástica no vetor de velocidades (se ativado pelo estado)
11 $Solucao_i \leftarrow$ ConstrucaoGuiada$(Escore_j)$
12 **if** *Probabilidade de Busca Local atingida* **then**
13 $Solucao_i \leftarrow$ BuscaLocalVND$(Solucao_i)$
14 Atualizar memória cognitiva ($pbest_i$) se $f(Solucao_i) > f(pbest_i)$
15 Atualizar memória social ($gbest$) e métrica de estagnação se houver melhoria no enxame
16 **return** *Melhor solução global encontrada ($gbest$)*

O Algoritmo 4.13 descreve o fluxo processual do AR-BPSO. Na linha 1, a população inicial de partículas é gerada utilizando métodos construtivos. Durante o laço principal (linha 2), o algoritmo monitora a taxa de estagnação em relação à melhor solução global ($gbest$) na linha 3 e ajusta o estado comportamental do enxame na linha 4. Para cada partícula em paralelo (linha 5), as velocidades contínuas são atualizadas (linha 7), limitadas ao intervalo $[-v_{\max}, v_{\max}]$ (linha 8) e convertidas em escores de prioridade via função sigmoide (linha 9). Após a eventual aplicação de perturbações estocásticas (linha 10), uma nova solução factível, denotada por $Solucao_i$ e correspondente ao vetor de posição x_i da formulação clássica do enxame, é construída de forma guiada pelos escores (linha 11) e pode ser refinada pela busca local VND conforme a probabilidade ativa (linha 13). Por fim, a memória cognitiva individual ($pbest_i$) é atualizada na linha 14, e a melhor solução social ($gbest$) é sincronizada ao final da iteração (linha 15).

4.6 Critérios de Parada e Calibração de Parâmetros

O critério de parada principal adotado neste trabalho segue as especificações estabelecidas pelo desafio SBPO 2025 (Mercado Livre, 2025), definindo o tempo limite de execução (*timeout*) de 600 segundos para cada instância avaliada.

No que tange às configurações operacionais dos algoritmos, o R-GRASP executa até

o limite de 2.000 iterações independentes, teto empírico estabelecido para viabilizar a convergência da probabilidade reativa em instâncias de grande porte sem ultrapassar o tempo limite de execução. Por sua vez, o AR-BPSO opera com uma população de 100 partículas ao longo de até 1.000 gerações, dimensionamento escolhido para equilibrar a diversidade populacional e o custo computacional por geração. Adicionalmente ao limite temporal absoluto, o AR-BPSO adota um critério de parada complementar por estagnação, encerrando a busca caso a melhor solução global permaneça por 150 gerações consecutivas sem aprimoramento na função objetivo. Tais parâmetros foram definidos para propiciar a exploração do espaço de busca, permitindo que ambas as metaheurísticas compitam sob restrições equivalentes de tempo computacional.

4.6.1 Definição Experimental dos Parâmetros e Padrões da Literatura

Devido a restrições computacionais e de escopo, a sintonia fina dos hiperparâmetros operacionais do *OOSP-MetaSolver Framework*, tais como o expoente de reatividade $q = 6$ e o fator de suavização da média móvel exponencial de 0,20, não decorreu de um *Design of Experiments* (DoE) formal com análise de variância exaustiva. Em vez disso, adotou-se uma calibração manual exploratória conduzida sobre um subconjunto reduzido de instâncias representativas do problema. Os valores finais selecionados apresentaram desempenho estável e consistência na convergência ao longo dos experimentos preliminares e, por esse motivo, foram mantidos em todas as avaliações apresentadas neste trabalho.

Adicionalmente, grande parte dos valores operacionais foi utilizada por ser o padrão consolidado na literatura científica ou por seguir heurísticas proporcionais de escala:

- **Critério de Expiração Tabu (τ):** Definida pela regra adaptativa de escala $\tau = \min(15, \lfloor\sqrt{|I|}\rfloor)$, que ajusta a memória restritiva de acordo com o tamanho da instância ($|I|$), impondo um teto de 15 iterações para evitar o bloqueio excessivo da exploração em instâncias de maior porte.
- **Memória do LAHC (L):** O comprimento do vetor de memória fixado em $L = 50$ segue o valor frequentemente adotado na literatura para o *Late Acceptance Hill Climbing* (Burke; Bykov, 2017), apresentando equilíbrio entre capacidade de transposição de platôs e convergência.
- **Fator de Aleatoriedade (α):** O intervalo $\alpha \in [0,05, 0,30]$ adotado nas construções randomizadas do R-GRASP encontra-se dentro da faixa normalmente utilizada em procedimentos semigulosos, favorecendo a diversidade populacional sem comprometer a qualidade da solução inicial.

5 Resultados Computacionais

Este capítulo apresenta as análises e os resultados computacionais obtidos na avaliação do *framework* proposto. Inicialmente, são descritos o ambiente de testes e o protocolo experimental na Seção 5.1; em seguida, é realizada a avaliação isolada das heurísticas construtivas responsáveis pela geração da solução inicial na Seção 5.2; posteriormente, as estruturas de busca local são analisadas quanto à sua capacidade de refinamento na Seção 5.3; logo após, avalia-se o desempenho das metaheurísticas GRASP Reativo e AR-BPSO na Seção 5.4; na Seção 5.5, estabelece-se o comparativo com as abordagens do estado da arte presentes na literatura; na Seção 5.6, analisa-se a evolução quantitativa incremental do pipeline algorítmico; por fim, a Seção 5.7 apresenta a síntese dos resultados e as diretrizes operacionais para o problema.

5.1 Ambiente de Testes e Protocolo Experimental

Nesta seção, são detalhadas as especificações do ambiente computacional utilizado para a condução dos experimentos, as características intrínsecas aos conjuntos de instâncias avaliados e a metodologia empregada para a aferição do desempenho dos algoritmos desenvolvidos.

5.1.1 Configuração Computacional

Todos os experimentos foram conduzidos em um ambiente computacional padronizado, de modo a permitir a reprodutibilidade e a confiabilidade das métricas de tempo e desempenho. O *hardware* utilizado é composto por um processador 12th Gen Intel® Core™ i5-12450H e 32 GB de memória RAM. O sistema operacional base foi a distribuição Linux Manjaro 24, com ambiente de trabalho KDE Plasma 6 e kernel 6.12.

Do ponto de vista de *software*, todo o *framework* algorítmico, bem como os módulos de avaliação, foram integralmente desenvolvidos na linguagem de programação Rust, compilados nativamente por meio do compilador `rustc 1.85+` em modo *release* para reduzir o tempo de execução.

Em razão da natureza estocástica e da quantidade de iterações requerida pelas metaheurísticas desenvolvidas, aplicou-se paralelismo de dados em nível de *thread* por meio da biblioteca `rayon`. Essa abordagem permitiu o processamento simultâneo e assíncrono tanto do conjunto de 100 partículas do enxame (AR-BPSO) quanto dos blocos de iterações independentes exigidos na fase de construção do GRASP. Além disso, os recursos de abstração de custo zero (*zero-cost abstractions*) do Rust possibilitaram que um grande número de soluções vizinhas fosse avaliado e descartado rapidamente, o que é necessário para manter a viabilidade computacional.

5.1.2 Conjunto de Instâncias

O conjunto de dados empregado na validação dos algoritmos provém do cenário real de e-commerce e logística fornecido no desafio acadêmico do Mercado Livre no LVII Simpósio Brasileiro de Pesquisa Operacional (Mercado Livre, 2025), disponível publicamente no repositório oficial do desafio (<https://github.com/mercadolibre/challenge-sbpo-2025>). Tais instâncias encapsulam a complexidade típica de centros de distribuição, englobando a dinâmica restritiva de múltiplos pedidos, capacidade fracionada de corredores e alocação dispersa de estoque de itens.

As instâncias foram divididas em três conjuntos, refletindo tanto as etapas metodológicas do desafio competitivo quanto uma progressão geral de escala e complexidade combinatória:

- **Conjunto A (Calibração e Linha de Base):** Composto por 20 instâncias cujos resultados de referência foram disponibilizados previamente pela organização para calibração dos modelos. Apresenta o menor porte médio, variando desde cenários triviais até instâncias com alguns milhares de itens, como em A06 e A14, caracterizando um espaço amostral diversificado que permitiu a aferição preliminar da convergência algorítmica e a sintonia dos parâmetros.
- **Conjunto B (Escalabilidade):** Consiste em 15 instâncias liberadas durante o desafio que elevam o porte do problema. Algumas instâncias deste conjunto chegam a abranger um considerável volume de pedidos simultâneos, mais de 480 corredores logísticos e acima de 53.000 itens totais, como na instância B11, demandando gestão eficiente de memória e estruturas de dados adequadas para manter o processamento.
- **Conjunto X (Teste Cego e Estresse Combinatório):** Formado por 15 instâncias destinadas aos testes cegos na etapa final da competição, cujas melhores soluções conhecidas (*Best Known Solutions*, BKS) foram reveladas *a posteriori*. Além de conter os cenários de maior escala absoluta, como a instância X14 com mais de 79.000 itens demandados, este conjunto destaca-se pela alta densidade combinatória. O arranjo logístico é concebido para forçar as abordagens a operarem sob sobreposição e disputa por corredores, servindo como teste de estresse da capacidade de exploração e diversificação das metaheurísticas em matrizes densas.

5.1.3 Metodologia de Avaliação

O protocolo de testes foi estruturado para avaliar a consistência, a estabilidade e o desempenho comparativo dos métodos. Levando em consideração a limitação imposta no desafio do Mercado Livre, todos os experimentos foram limitados a 600 segundos (10 minutos) de tempo de execução por instância. Os parâmetros nominais dos algoritmos operam com limites de 2.000 iterações para o R-GRASP e 1.000 gerações para o AR-BPSO, valores definidos em testes preliminares para favorecer a convergência em cenários sem limitação temporal. Adici-

onalmente, o AR-BPSO conta com um critério de parada complementar acionado após 150 gerações consecutivas sem melhoria, calibrado empiricamente para interromper estagnações prolongadas. Apesar dessas configurações estruturais, o rigor do protocolo experimental impôs uma interrupção igualitária entre os métodos, paralisando a execução de todas as rodadas no limite de 600 segundos.

Em função da natureza estocástica inerente às abordagens desenvolvidas, cada instância foi executada independentemente 10 vezes com diferentes sementes aleatórias para obter robustez nas métricas e mitigar eventuais enviesamentos por ótimos locais.

Para analisar quantitativamente o desempenho alcançado pelas soluções, foram consolidadas as seguintes métricas a partir dos dados gerados:

- Melhor pontuação (Best): Refere-se ao maior valor absoluto da função objetivo obtido durante as 10 execuções, indicando a capacidade da metaheurística de encontrar soluções de alta qualidade no espaço de busca.
- Média (Avg): Consiste na média aritmética dos valores encontrados ao final das 10 repetições, operando como métrica primária para mensurar a consistência e a estabilidade convergencial do algoritmo.
- Tempo de Execução (T(s)): Indica a média temporal em segundos demandada pelo algoritmo para alcançar e registrar o seu melhor resultado antes da ocorrência do limite de corte, avaliando a celeridade da otimização.
- Gap para a *Best Known Solution* (BKS): Representa a diferença percentual entre o melhor valor da função objetivo obtido pelo algoritmo (Best) e a melhor solução conhecida (BKS) para a respectiva instância. Esses valores de referência foram disponibilizados integralmente pela organização oficial do desafio SBPO 2025 (Mercado Livre, 2025). A métrica é calculada de acordo com a Equação 5.1:

$$Gap = \frac{BKS - Best}{BKS} \tag{5.1}$$

5.2 Avaliação e Comparação das Heurísticas Construtivas

Nesta seção, é avaliado o desempenho isolado das abordagens de construção inicial da solução, antes da aplicação de qualquer etapa de busca local ou metaheurística. O objetivo primário consiste em ilustrar como cada estratégia gulosamente constrói a rota e atende aos pedidos, identificando quais delas oferecem o melhor ponto de partida para o refinamento subsequente.

A Tabela 5.1 sumariza os resultados obtidos. As colunas da tabela caracterizam as especificações das instâncias e os desvios alcançados por cada abordagem: a coluna Inst. identifica

a instância testada; Ped. indica o número total de pedidos; Cor. apresenta a quantidade total de corredores do armazém; Itens Totais denota a quantidade acumulada de itens de todos os pedidos demandados; e BKS representa o valor da melhor solução conhecida documentada na literatura para comparação. Estruturalmente, a tabela, assim como as demais tabelas de consolidação deste capítulo, está dividida em três blocos horizontais que segregam as instâncias de acordo com os Conjuntos A, B e X, conforme detalhado na Seção 5.1.

As colunas de resultados comparam seis variações construtivas, divididas nas duas orientações principais. As estratégias orientadas a pedido são representadas pela versão estática determinística OC (*Order-Centric*), pela versão gulosa adaptativa com recálculo dinâmico OC-A e pela versão aleatorizada com Lista Restrita de Candidatos OC-R. Por sua vez, as estratégias orientadas a corredor são representadas pela versão estática determinística AC (*Aisle-Centric*), pela versão gulosa adaptativa com recálculo dinâmico AC-A e pela versão aleatorizada com Lista Restrita de Candidatos AC-R. Para cada variação, são reportados o valor médio final da função objetivo alcançado ao longo das execuções independentes (Avg) e o menor desvio percentual residual em relação à BKS, correspondendo à melhor solução encontrada entre as repetições (Gap).

Tabela 5.1 – Avaliação das Heurísticas Construtivas

Inst.	Ped.	Cor.	Itens Totais	BKS	OC		OC-A		OC-R		AC		AC-A		AC-R	
					Avg	Gap	Avg	Gap	Avg	Gap	Avg	Gap	Avg	Gap	Avg	Gap
A01	61	116	226	15,00	7,63	32,00%	7,46	43,33%	7,01	48,33%	9,40	37,33%	15,00	**0,00%**	13,25	**0,00%**
A02	7	33	11	2,00	2,00	**0,00%**	2,00	**0,00%**	2,00	**0,00%**	2,00	**0,00%**	2,00	**0,00%**	2,00	**0,00%**
A03	82	124	344	12,00	7,67	29,76%	8,21	10,42%	6,88	34,72%	5,33	55,56%	6,22	48,15%	6,84	40,00%
A04	16	91	71	3,50	3,50	**0,00%**	3,50	**0,00%**	3,50	**0,00%**	1,38	60,71%	2,33	33,33%	2,04	33,33%
A05	2625	161	13813	177,88	75,40	56,65%	80,89	51,97%	74,59	55,90%	171,67	3,49%	150,54	15,37%	150,55	13,23%
A06	10341	184	14732	691,00	130,31	80,44%	142,42	77,80%	132,71	79,76%	310,50	55,07%	691,00	**0,00%**	654,10	**0,00%**
A07	8320	180	10885	392,25	41,81	88,98%	99,68	71,02%	47,65	87,26%	196,43	49,92%	392,25	**0,00%**	387,50	0,25%
A08	2185	168	13034	162,94	50,82	67,73%	105,06	24,75%	52,02	65,82%	138,94	14,73%	153,17	6,00%	151,85	4,06%
A09	70	304	279	4,42	3,19	21,89%	3,23	15,90%	2,85	32,76%	1,45	67,27%	2,30	47,99%	2,15	48,09%
A10	1602	383	4645	17,11	7,40	54,29%	7,89	51,03%	6,77	57,47%	11,33	33,78%	13,05	23,72%	12,98	23,37%
A11	1029	375	3632	16,85	9,04	41,98%	10,67	29,21%	7,69	52,04%	9,66	42,66%	11,22	33,44%	11,20	32,62%
A12	133	342	525	11,25	6,64	33,33%	6,64	30,37%	6,01	34,22%	2,27	79,81%	3,45	69,29%	3,99	60,00%
A13	8375	413	9905	117,38	18,06	84,19%	31,58	72,47%	18,93	83,24%	66,12	43,67%	117,08	0,26%	115,70	0,46%
A14	12402	413	14980	181,64	26,48	85,10%	45,76	73,19%	27,50	84,62%	104,19	42,64%	181,36	0,15%	179,78	0,50%
A15	7367	402	9008	149,33	28,97	79,98%	31,36	76,32%	28,55	80,18%	102,86	31,12%	148,33	0,67%	147,07	**0,00%**
A16	1108	88	1108	85,00	12,11	85,05%	22,73	67,45%	18,43	75,20%	75,50	11,18%	85,00	**0,00%**	81,75	**0,00%**
A17	417	83	417	36,50	4,95	85,47%	11,73	63,01%	8,20	75,80%	15,88	56,51%	36,50	**0,00%**	35,90	**0,00%**
A18	2682	90	3105	117,20	18,61	83,42%	55,11	48,24%	23,00	79,02%	85,50	27,05%	116,80	0,34%	113,26	1,88%
A19	2257	134	2772	202,00	33,57	82,25%	39,12	78,51%	33,34	82,52%	202,00	**0,00%**	202,00	**0,00%**	202,00	**0,00%**
A20	5	5	15	5,00	3,68	10,00%	3,72	10,00%	3,67	**0,00%**	4,50	10,00%	4,50	10,00%	4,70	**0,00%**
Médias A					**58,54%**	**55,13%**	**50,58%**	**44,75%**	**59,18%**	**55,44%**	**36,12%**	**36,12%**	**14,44%**	**14,44%**	**15,87%**	**12,89%**
B01	11321	185	14973	631,50	54,38	90,48%	125,39	78,31%	60,13	89,87%	329,36	47,84%	631,50	**0,00%**	558,93	**0,00%**
B02	2162	165	14365	180,64	97,64	43,97%	79,24	43,77%	73,42	50,97%	143,59	20,51%	169,46	6,19%	165,48	4,37%
B03	2070	164	13013	149,60	80,08	44,31%	91,01	28,81%	68,25	46,67%	110,42	26,19%	136,92	8,48%	135,45	7,09%
B04	2448	163	14195	155,94	74,08	48,43%	104,64	28,27%	67,51	50,21%	133,43	14,44%	144,95	7,05%	146,52	3,06%
B05	6468	179	9489	368,00	52,04	85,01%	110,38	68,99%	59,59	83,05%	207,08	43,73%	360,00	2,17%	363,03	**0,00%**
B06	1857	165	9380	84,03	45,72	43,04%	45,23	39,31%	39,97	48,50%	73,28	12,79%	74,84	10,93%	75,27	7,38%
B07	1841	165	2331	108,00	25,82	74,49%	25,70	67,59%	26,29	73,41%	52,00	51,85%	108,00	**0,00%**	108,00	**0,00%**

Tabela 5.1 – Continuação

Inst.	Ped.	Cor.	Itens Totais	BKS	OC		OC-A		OC-R		AC		AC-A		AC-R	
					Avg	Gap	Avg	Gap	Avg	Gap	Avg	Gap	Avg	Gap	Avg	Gap
B08	12334	398	14979	227,10	22,40	89,94%	47,41	77,42%	23,75	89,22%	177,54	21,82%	225,50	0,70%	217,98	1,54%
B09	5581	417	7320	82,50	17,17	78,42%	26,52	66,56%	21,49	73,21%	45,00	45,45%	82,29	0,26%	80,61	0,78%
B10	14952	482	18870	444,00	50,17	88,51%	57,73	85,99%	50,59	88,34%	427,33	3,75%	444,00	**0,00%**	415,10	3,83%
B11	45112	482	53747	898,80	42,89	95,20%	142,25	83,71%	44,88	94,96%	865,20	3,74%	893,10	0,63%	878,14	1,47%
B12	28263	483	31451	567,64	28,31	94,97%	84,42	84,88%	31,25	94,39%	529,50	6,72%	567,18	0,08%	534,92	2,37%
B13	38214	483	49582	757,00	48,13	93,52%	130,76	82,33%	53,87	92,75%	703,70	7,04%	746,89	1,34%	726,71	1,47%
B14	38202	483	49152	885,17	60,79	92,97%	139,08	83,85%	63,17	92,63%	865,17	2,26%	878,00	0,81%	844,22	2,35%
B15	11541	398	14981	166,14	28,05	82,79%	48,17	70,06%	33,62	79,24%	116,20	30,06%	160,80	3,22%	157,19	4,26%
Médias B					**77,47%**	**76,40%**	**69,47%**	**65,99%**	**78,66%**	**76,49%**	**22,55%**	**22,55%**	**2,79%**	**2,79%**	**5,47%**	**2,66%**
X01	1949	164	7362	70,85	30,50	55,21%	36,38	46,11%	34,81	47,51%	59,91	15,43%	61,91	12,61%	61,01	11,82%
X02	2942	168	3992	217,67	33,78	83,53%	41,28	79,17%	37,70	80,75%	127,50	41,42%	217,67	**0,00%**	212,20	**0,00%**
X03	11330	180	14964	667,67	43,43	93,26%	150,00	75,90%	49,70	92,30%	314,25	52,93%	667,33	0,05%	657,13	0,05%
X04	2732	172	14471	202,75	112,65	35,30%	96,97	49,90%	79,06	52,19%	183,00	9,74%	185,16	8,68%	182,37	8,31%
X05	11024	182	14911	628,67	58,44	90,34%	131,24	76,82%	67,65	88,89%	325,12	48,28%	628,67	**0,00%**	617,57	0,21%
X06	2704	168	14105	191,86	55,48	68,95%	97,88	40,87%	54,22	70,15%	179,12	6,64%	176,00	8,27%	178,51	4,30%
X07	8725	418	11079	115,15	24,12	78,61%	35,22	68,29%	31,23	72,11%	62,02	46,14%	112,00	2,74%	111,04	2,74%
X08	20364	482	24761	438,22	27,75	93,49%	67,03	84,21%	33,18	92,28%	398,60	9,04%	419,70	4,23%	411,11	4,93%
X09	39701	483	50179	819,88	48,19	94,01%	130,19	83,54%	49,49	93,85%	791,88	3,42%	813,00	0,84%	790,16	2,94%
X10	44193	482	54944	964,62	34,33	96,39%	141,81	84,67%	34,97	96,34%	914,44	5,20%	960,12	0,47%	906,29	5,62%
X11	38657	483	47957	1060,00	120,11	88,40%	153,72	84,71%	119,50	88,61%	1003,50	5,33%	1031,00	2,74%	989,85	2,08%
X12	52322	483	61503	1339,67	98,34	92,30%	184,10	85,14%	98,26	92,46%	1281,33	4,35%	1333,00	0,50%	1277,90	2,71%
X13	49992	483	57285	1060,71	65,21	93,79%	152,65	85,19%	67,02	93,61%	1037,57	2,18%	1056,00	0,44%	1020,93	2,30%
X14	68064	483	79715	1633,67	71,04	95,49%	194,08	87,52%	79,79	95,01%	1569,67	3,92%	1625,67	0,49%	1544,33	4,45%
X15	2839	171	14688	263,91	82,59	67,09%	143,84	38,49%	80,74	67,10%	245,33	7,04%	238,67	9,56%	239,68	6,79%
Médias X					**82,93%**	**81,74%**	**73,47%**	**71,37%**	**82,89%**	**81,54%**	**17,41%**	**17,40%**	**3,44%**	**3,44%**	**5,72%**	**3,95%**
Médias Globais					**71,53%**	**69,49%**	**63,11%**	**59,11%**	**72,14%**	**69,59%**	**26,43%**	**26,43%**	**7,65%**	**7,64%**	**9,71%**	**7,14%**

5.2.1 Desempenho Isolado: Order-Centric vs. Aisle-Centric

As heurísticas construtivas propostas operam em duas variantes lógicas: orientadas a pedido (OC) e orientadas a corredor (AC). Os resultados consolidados na Tabela 5.1 apontam que o desempenho comparativo entre as estratégias está diretamente condicionado à razão de densidade combinatória da instância, expressa pela proporção entre o número total de pedidos demandados e a quantidade total de corredores disponíveis no armazém ($R = \frac{\text{Ped.}}{\text{Cor.}}$).

Em cenários operacionais onde o número de pedidos é inferior à capacidade de corredores do centro de distribuição ($R < 1$), as estratégias orientadas a pedido alcançam menores desvios residuais. Nas instâncias A02 ($R = 0,21$), A04 ($R = 0,18$), A09 ($R = 0,23$) e A12 ($R = 0,39$), as variações OC e OC-A superam as abordagens baseadas em corredor, alcançando gaps entre 0,00% e 30,37%, enquanto as variações AC registram desvios entre 33,33% e 60,00%. Nesse regime de baixa concorrência por recursos ($R < 1$), onde a demanda se encontra dispersa no armazém, priorizar o fechamento integral de pedidos esparsos evita a ativação desnecessária de corredores que conteriam apenas frações de outros pedidos e não gerariam incremento imediato na função objetivo.

Por outro lado, quando a densidade combinatória ultrapassa o limiar unitário ($R \geq 1$), a relação de desempenho se inverte de forma sistemática. Em 44 das 50 instâncias avaliadas onde a razão é igual ou superior a 1, englobando a totalidade dos cenários de grande escala do Conjunto B e de estresse combinatório do Conjunto X, as heurísticas orientadas a corredor (AC) dominam a comparação, obtendo maior pontuação média final em todas as 44 instâncias e menor desvio percentual em 42 delas. Na instância B11, onde a demanda abrange 45.112 pedidos para 484 corredores ($R = 93,21$), a construtiva OC estática obtém um desvio de 95,20%, enquanto a construtiva AC estática atinge 3,74% antes de qualquer procedimento de melhoria. Nessa estratégia, tentar atender a pedidos de forma isolada pulveriza as coletas e exaure rapidamente o limite de ativação de corredores. Em contrapartida, a consolidação por corredor agrupa a coleta de milhares de itens dentro das prateleiras já ativadas, centralizando o fluxo logístico e maximizando a sobreposição do estoque.

Consequentemente, a razão $R = \frac{\text{Ped.}}{\text{Cor.}}$ atua como um critério analítico prático para a seleção da heurística construtiva inicial no planejamento logístico: quando $R < 1$, a construção deve ser orientada a pedidos; quando $R > 1$, a construção deve ser orientada a corredores.

5.2.2 Impacto do Recálculo Adaptativo e da Aleatorização

Além das abordagens determinísticas estáticas OC e AC, que ordenam os candidatos uma única vez no início, foram avaliadas variações que incorporam o recálculo dinâmico do *score* de prioridade, que estima o impacto do candidato na função objetivo a cada passo construtivo, representadas pelas variações adaptativas gulosas OC-A e AC-A, e variações estocásticas com Lista Restrita de Candidatos, indicadas por OC-R e AC-R.

A transição de uma lista estática para a seleção adaptativa revelou-se relevante para reduzir escolhas subótimas causadas por informações desatualizadas. Na instância X14, por exemplo, a versão estática orientada a corredor obteve um gap de 3,92%. Ao permitir que o algoritmo recalcule dinamicamente a utilidade marginal dos corredores a cada passo na versão AC-A, o gap da solução inicial foi reduzido para 0,49%. Esse resultado indica que atualizar o *score* de prioridade com base nos itens já consolidados ou no custo marginal dos novos corredores permite montar rotas com menor desvio residual que a ordem estática predefinida.

Na variante orientada a pedidos, a incorporação de adaptatividade também confere melhorias sobre o modelo estático. A transição da avaliação determinística (OC) para o recálculo dinâmico (OC-A) reduz o desvio percentual médio global de 69,49% para 59,11%, indicando que atualizar a utilidade dos pedidos restantes a cada passo mitiga alocações subótimas. Em instâncias como A08, o recálculo adaptativo reduz o gap de 67,73% para 24,75%. Contudo, em conformidade com o critério da razão $R = \frac{\text{Ped.}}{\text{Cor.}}$, quando a instância opera sob regime de alta concorrência de prateleiras ($R \geq 1$), o recálculo dinâmico em OC-A e a aleatorização em OC-R, embora melhorem a pontuação em relação ao OC estático, não são suficientes para igualar o desempenho da abordagem baseada em corredor. Nas instâncias do Conjunto B ($R \gg 1$), as

variações OC-A e OC-R mantiveram gaps entre 43,77% e 89,87%, apontando que nessa estratégia a limitação primária reside na própria lógica estrutural de roteamento por pedido, e não no caráter estático ou dinâmico de sua seleção.

Cabe esclarecer que, embora as abordagens estáticas (OC, AC) e adaptativas (OC-A, AC-A) sejam nominalmente determinísticas em sua lógica de seleção gulosa, os dados consolidados na Tabela 5.1 registram variabilidade ao longo das execuções independentes. Esse comportamento decorre do pré-processamento na fase de leitura de entrada em Rust. A determinação dos corredores necessários para o atendimento de cada pedido emprega coleções de mapeamento (`HashMap` e `HashSet`), que implementam hashing de segurança com semente aleatória gerada por processo. Em situações de empate, onde múltiplos corredores fornecem idêntica cobertura de itens para um mesmo pedido, o critério de seleção depende diretamente da ordem interna de iteração dessas estruturas. Desse modo, a alteração da semente a cada inicialização modifica a matriz pré-processada de corredores requeridos, induzindo caminhos de exploração distintos e justificando as variações observadas nas heurísticas sem Lista Restrita de Candidatos.

5.2.3 Síntese e Comparação Geral dos Construtivos

A análise consolidada das 50 instâncias avaliadas fundamenta os critérios que justificam a escolha das heurísticas construtivas base para as etapas posteriores do algoritmo. Como 88% do conjunto experimental (44 de 50 instâncias) opera com uma característica de alta densidade combinatória ($R \geq 1$), a análise global das métricas aponta uma superioridade das abordagens orientadas a corredor em relação às orientadas a pedido.

Enquanto a abordagem estática orientada a pedidos (OC) apresentou um desvio médio global de 69,49% em relação à BKS e a variação adaptativa (OC-A) alcançou 59,11%, a heurística orientada a corredor estática (AC) registrou desvio médio de 26,44%. A introdução do recálculo adaptativo na variação AC-A reduziu essa média global para 7,64%, e o controle probabilístico na variação AC-R obteve 7,14% em seu pico de melhor resultado. Contudo, é imperativo observar o comportamento médio dessas abordagens: enquanto o AC-A entregou um desvio médio de 7,65% por execução, o AC-R registrou uma média geral de 9,71%, atestando a alta variância introduzida pela lista restrita de candidatos. Devido a essa previsibilidade determinística e melhor consistência na média, o AC-A apresentou-se como a heurística mais confiável e robusta para alimentar as etapas posteriores de refinamento. Nos cenários de maior porte e complexidade, como o Conjunto B ($R \gg 1$), o desvio médio de AC-A atingiu 2,79%, e no Conjunto X registrou 3,44%, confirmando a eficácia da consolidação dinâmica de prateleiras na geração de rotas iniciais consistentes.

Por outro lado, o desvio percentual médio mais elevado de AC-A e AC-R no Conjunto A decorre da presença das instâncias com razão de densidade $R < 1$ nesse subconjunto, a exemplo de A09 ($R = 0,23$) e A12 ($R = 0,39$), onde o roteamento por corredor apresenta estruturalmente desempenho inferior ao por pedido, somada à sensibilidade matemática das

métricas em cenários de menor porte. Em instâncias com valores de função objetivo baixos, variações de um ou dois pontos na coleta geram impactos percentuais acentuados no cálculo do gap, o que eleva os resultados médios desse subconjunto em comparação com os conjuntos de grande escala onde o erro residual se dilui.

A eficácia do recálculo adaptativo (AC-A) e da Lista Restrita de Candidatos (AC-R) também se constata na quantidade de soluções ótimas encontradas diretamente durante a fase construtiva. Entre as 50 instâncias avaliadas, as variações AC-A e AC-R alcançaram de forma independente o valor ótimo conhecido ($0,00\%$ de gap) em 12 instâncias, correspondendo a 24,00% do conjunto experimental total. Essa convergência direta distribuiu-se em 7 instâncias do Conjunto A (35,00% do subconjunto), 3 instâncias do Conjunto B (20,00% do subconjunto) e 2 instâncias do Conjunto X (13,33% do subconjunto), indicando que, em instâncias onde a razão combinatória é elevada ($R = \frac{\text{Ped.}}{\text{Cor.}} \geq 1$), a consolidação por corredor maximiza a sobreposição de estoque nas prateleiras já ativadas e favorece atingir a fronteira ótima sem necessidade de busca local.

Com base nesse diagnóstico, as heurísticas orientadas a corredor adaptativas e randomizadas (AC-A e AC-R) configuram a estratégia padrão de construção inicial para as metaheurísticas posteriores (GRASP e BPSO). Ao iniciar a otimização com rotas estruturalmente consolidadas nos corredores para a expressiva maioria dos cenários de alta densidade, o esforço computacional das fases de melhoria é utilizado para o aperfeiçoamento fino das alocações e dos movimentos locais da vizinhança.

5.3 Análise e Comparação das Estruturas de Buscas Locais e Metaheurísticas Clássicas

Nesta seção, é avaliada a capacidade de refinamento das diferentes estratégias de busca local aplicadas sobre as soluções iniciais. O objetivo principal consiste em isolar a etapa de busca local, a fim de avaliar o impacto dos movimentos de vizinhança na exploração do espaço de busca, bem como analisar a relação entre ganho e tempo computacional exigida para processar essas melhorias.

Dentre todas as estratégias testadas, a Tabela 5.2 consolida os resultados das três abordagens selecionadas para comparação direta: os movimentos de trajetória única, representados pela Busca Tabu e pelo *Late Acceptance Hill Climbing* (LAHC), e o procedimento estruturado do *Variable Neighborhood Descent* (VND). As cinco primeiras colunas caracterizam a instância testada: Inst. identifica o nome da instância; Ped. indica o número total de pedidos; Cor. informa a quantidade de corredores do armazém; Itens Totais expressa a soma de todos os itens demandados; e BKS representa o valor da melhor solução conhecida documentada na literatura para comparação. Para cada um dos três métodos avaliados, VND, Tabu e LAHC, a tabela apresenta quatro colunas de métricas: Avg indica o valor médio final da função objetivo alcançado pelo

método nas execuções; Ganho denota o incremento médio obtido na função objetivo em relação à solução inicial construída; Gap reporta o menor desvio percentual em relação à BKS, correspondendo à melhor solução encontrada pelo método entre as repetições; e T(s) expressa o tempo computacional médio de processamento em segundos.

Tabela 5.2 – Avaliação da Busca Local e das Metaheurísticas Clássicas

Inst.	Ped.	Cor.	Itens Totais	BKS	VND				Tabu				LAHC			
					Avg	Ganho	Gap	T(s)	Avg	Ganho	Gap	T(s)	Avg	Ganho	Gap	T(s)
A01	61	116	226	15,00	12,20	0,83	**0,00%**	0,00	11,97	0,36	**0,00%**	0,13	11,85	0,29	**0,00%**	0,00
A02	7	33	11	2,00	2,00	0,00	**0,00%**	0,00	2,00	0,00	**0,00%**	0,01	2,00	0,00	**0,00%**	0,00
A03	82	124	344	12,00	10,16	2,95	2,78%	0,00	10,59	3,42	2,78%	0,20	9,21	2,03	2,78%	0,00
A04	16	91	71	3,50	3,30	0,39	**0,00%**	0,00	3,50	0,58	**0,00%**	0,02	3,50	0,58	**0,00%**	0,00
A05	2625	161	13813	177,88	148,49	32,13	4,99%	0,21	126,91	9,64	15,37%	2,73	117,76	1,69	15,33%	0,02
A06	10341	184	14732	691,00	595,50	177,89	**0,00%**	0,11	434,57	19,03	**0,00%**	0,08	415,49	0,10	**0,00%**	0,02
A07	8320	180	10885	392,25	358,26	112,38	**0,00%**	0,21	255,69	8,67	**0,00%**	0,07	246,79	0,09	**0,00%**	0,02
A08	2185	168	13034	162,94	144,36	16,61	6,00%	0,11	135,82	7,09	6,00%	2,91	130,37	1,71	6,00%	0,01
A09	70	304	279	4,42	4,05	1,25	**0,00%**	0,01	4,20	1,48	**0,00%**	0,24	3,75	1,01	**0,00%**	0,00
A10	1602	383	4645	17,11	13,69	3,13	13,40%	0,09	12,32	1,79	22,11%	2,17	10,55	0,20	23,59%	0,01
A11	1029	375	3632	16,85	14,46	3,60	3,91%	0,08	13,47	2,64	9,90%	2,28	11,45	0,46	25,49%	0,01
A12	133	342	525	11,25	9,21	4,24	**0,00%**	0,01	9,79	4,70	8,89%	0,26	8,22	3,02	8,89%	0,00
A13	8375	413	9905	117,38	99,50	25,12	0,26%	0,13	75,68	1,48	0,26%	0,08	74,45	0,05	0,26%	0,02
A14	12402	413	14980	181,64	152,90	39,60	0,15%	0,27	115,27	2,01	0,15%	0,11	113,39	0,01	0,15%	0,04
A15	7367	402	9008	149,33	123,17	33,27	0,67%	0,08	91,66	1,85	0,67%	0,06	89,22	0,01	0,67%	0,02
A16	1108	88	1108	85,00	77,12	23,35	**0,00%**	0,01	53,99	0,00	**0,00%**	0,88	53,64	0,00	**0,00%**	0,00
A17	417	83	417	36,50	34,30	10,50	**0,00%**	0,01	24,29	0,60	**0,00%**	0,78	23,80	0,00	**0,00%**	0,00
A18	2682	90	3105	117,20	108,49	23,15	0,34%	0,03	87,01	1,60	0,34%	1,05	86,14	0,54	0,34%	0,01
A19	2257	134	2772	202,00	202,00	79,25	**0,00%**	0,03	126,29	2,84	**0,00%**	1,15	125,92	0,78	**0,00%**	0,01
A20	5	5	15	5,00	4,58	0,26	**0,00%**	0,00	4,62	0,41	**0,00%**	0,01	4,60	0,38	**0,00%**	0,00
Médias A							1,62%	0,07			3,32%	0,76			4,17%	0,01
B01	11321	185	14973	631,50	561,75	184,08	**0,00%**	0,24	395,91	15,81	**0,00%**	0,09	379,38	0,08	**0,00%**	0,02
B02	2162	165	14365	180,64	161,70	37,99	2,01%	0,09	139,44	14,23	6,19%	3,13	125,74	1,75	6,19%	0,01
B03	2070	164	13013	149,60	140,87	24,39	**0,00%**	0,08	126,78	15,21	7,59%	3,18	111,47	2,87	8,48%	0,01
B04	2448	163	14195	155,94	147,74	23,14	2,87%	0,12	135,21	12,19	5,83%	3,07	127,63	3,03	7,05%	0,01
B05	6468	179	9489	368,00	354,57	119,28	1,90%	0,11	248,55	13,90	2,17%	0,06	236,12	0,20	2,17%	0,01
B06	1857	165	9380	84,03	74,56	15,00	3,25%	0,09	68,82	9,26	10,06%	2,78	61,01	1,22	10,93%	0,01
B07	1841	165	2331	108,00	87,97	21,36	**0,00%**	0,02	70,02	2,82	**0,00%**	1,01	67,28	1,24	**0,00%**	0,01
B08	12334	398	14979	227,10	196,04	59,47	0,70%	0,31	139,01	2,62	0,70%	0,11	135,97	0,06	0,70%	0,04
B09	5581	417	7320	82,50	72,55	18,08	0,26%	0,07	55,76	1,52	0,26%	0,06	54,56	0,20	0,26%	0,02
B10	14952	482	18870	444,00	391,80	141,01	**0,00%**	0,44	256,84	5,56	**0,00%**	0,13	251,25	0,05	**0,00%**	0,05
B11	45112	482	53747	898,80	764,57	247,90	0,63%	6,54	520,26	3,35	0,63%	0,65	516,28	0,03	0,63%	0,47
B12	28263	483	31451	567,64	491,08	164,25	0,08%	2,41	327,35	1,51	0,08%	0,37	326,70	0,01	0,08%	0,24
B13	38214	483	49582	757,00	653,35	214,93	1,34%	3,31	443,08	4,64	1,34%	0,45	437,88	0,06	1,34%	0,30
B14	38202	483	49152	885,17	804,96	297,26	0,81%	2,88	512,92	6,22	0,81%	0,39	506,60	0,06	0,81%	0,23
B15	11541	398	14981	166,14	145,33	40,93	3,22%	0,24	107,02	2,47	3,22%	0,10	104,07	0,02	3,22%	0,04
Médias B							1,14%	1,13			2,59%	1,04			2,79%	0,10
X01	1949	164	7362	70,85	61,46	12,37	5,80%	0,09	52,68	4,26	11,94%	2,25	50,21	0,97	12,61%	0,01
X02	2942	168	3992	217,67	188,80	59,39	**0,00%**	0,05	134,30	4,53	**0,00%**	1,19	130,18	0,76	**0,00%**	0,01
X03	11330	180	14964	667,67	596,36	187,56	0,05%	0,35	422,51	14,69	0,05%	0,09	407,55	0,24	0,05%	0,03
X04	2732	172	14471	202,75	179,20	38,28	6,75%	0,21	148,24	8,51	8,68%	3,40	142,74	1,97	8,68%	0,02
X05	11024	182	14911	628,67	553,03	173,73	**0,00%**	0,29	393,32	13,02	**0,00%**	0,09	378,38	0,20	**0,00%**	0,03
X06	2704	168	14105	191,86	165,85	29,84	8,27%	0,14	150,22	13,14	8,22%	3,27	139,55	2,33	8,27%	0,02

Tabela 5.2 – Continuação

Inst.	Ped.	Cor.	Itens Totais	BKS	VND				Tabu				LAHC			
					Avg	Ganho	Gap	T(s)	Avg	Ganho	Gap	T(s)	Avg	Ganho	Gap	T(s)
X07	8725	418	11079	115,15	100,67	26,44	2,74%	0,13	76,46	2,21	2,74%	0,08	74,48	0,03	2,74%	0,03
X08	20364	482	24761	438,22	361,67	118,03	4,23%	1,02	246,63	2,73	4,23%	0,22	243,55	0,02	4,23%	0,11
X09	39701	483	50179	819,88	723,38	251,87	0,84%	3,55	475,60	4,39	0,84%	0,48	471,91	0,02	0,84%	0,32
X10	44193	482	54944	964,62	830,93	279,75	0,47%	5,01	555,41	4,25	0,47%	0,56	549,57	0,04	0,47%	0,39
X11	38657	483	47957	1060,00	993,77	401,74	2,74%	2,07	608,37	12,83	2,74%	0,32	592,34	0,05	2,74%	0,15
X12	52322	483	61503	1339,67	1254,32	495,97	0,50%	4,27	764,30	7,82	0,50%	0,53	757,77	0,05	0,50%	0,31
X13	49992	483	57285	1060,71	929,08	324,33	0,44%	6,75	607,59	3,59	0,44%	0,66	605,14	0,02	0,44%	0,48
X14	68064	483	79715	1633,67	1532,75	621,61	0,49%	6,82	917,65	7,65	0,49%	0,71	910,96	0,03	0,49%	0,48
X15	2839	171	14688	263,91	241,05	51,69	2,43%	0,15	210,90	17,59	8,86%	3,11	191,54	3,36	9,56%	0,02
Médias X							2,38%	2,06			3,35%	1,13			3,44%	0,16
Médias Globais							1,71%	0,98			3,11%	0,96			3,54%	0,08

5.3.1 Influência da Solução Inicial no Refinamento (OC-A vs. AC-A)

Os resultados apresentados na Tabela 5.2 representam uma consolidação das execuções iniciadas por duas estratégias construtivas adaptativas gulosas: a heurística orientada a pedidos (OC-A) e a heurística orientada a corredores (AC-A). Para as métricas de tempo de execução (T(s)), ganho e valor médio final (Avg), os dados exibidos constituem a média aritmética do desempenho das buscas locais a partir de ambas as origens. O desvio percentual final (Gap), por sua vez, reporta o melhor valor encontrado entre as duas abordagens.

Essa agregação foi adotada para avaliar o comportamento estrutural dos operadores de busca local sob diferentes níveis de qualidade inicial. Avaliar o refinamento partindo exclusivamente de soluções de alta qualidade (AC-A) resultaria em ganhos marginais reduzidos, não refletindo a real capacidade de reestruturação das vizinhanças. Em contrapartida, utilizar apenas as soluções com maior desvio inicial (OC-A) inflaria excessivamente o ganho numérico absoluto, não condizendo com a aplicação prática das metaheurísticas finais, como o R-GRASP, que alternam probabilisticamente entre os construtivos e selecionam as melhores origens. Portanto, a média aritmética fornece um parâmetro balanceado para comparar as estratégias de refinamento.

Os resultados globais indicam o desempenho do VND em comparação aos métodos de trajetória única. O VND registrou o menor desvio percentual final da etapa, atingindo um gap médio de 1,71% em relação à BKS. A Busca Tabu e o LAHC apresentaram desvios globais de 3,11% e 3,54%, respectivamente. O custo computacional médio do VND foi de 0,98 segundos, exigindo tempo de execução análogo ao da Busca Tabu, que demandou 0,96 segundos, mas fornecendo rotas com menor erro percentual final.

5.3.2 Eficiência das Vizinhanças e Trajetórias Isoladas

A análise das estruturas isoladas visa compreender como as heurísticas de trajetória única contribuem para o incremento da função objetivo antes de atingirem o critério de parada

por ausência de melhora em suas respectivas vizinhanças. Os resultados sugerem que essas abordagens apresentam perfis de comportamento distintos no balanço entre custo e benefício computacional, mas ambas encontram dificuldades para reduzir o distanciamento para a fronteira ótima.

Observando as métricas, constata-se que a técnica baseada no critério de aceitação atrasada (LAHC) manteve um custo computacional reduzido, com apenas 0,08 segundos de tempo médio, resultando em um desvio percentual médio final de 3,54%. Por sua vez, a Busca Tabu apresentou maior capacidade exploratória ao despender um tempo de processamento médio de 0,96 segundos, indicando que sua lista de memória auxiliou levemente na navegação de espaços de busca e na superação de ótimos locais, reduzindo o desvio percentual médio final para 3,11%. Contudo, os números sugerem que, isoladas, as trajetórias únicas são insuficientes para lidar com as restrições da topologia logística.

5.3.3 Desempenho do VND

Para explorar as complementaridades das vizinhanças e buscar superar mínimos locais nos quais os movimentos de uma única estrutura de vizinhança tendem a estagnar, aplicou-se a estratégia do *Variable Neighborhood Descent* (VND). Este procedimento obteve os maiores ganhos de refinamento entre os métodos avaliados.

Os resultados mostram diferenças entre a aplicação de trajetórias isoladas e o VND. O VND atingiu o menor desvio percentual médio da etapa, reduzindo o gap para 1,71%. Em termos de custo computacional, o VND demandou tempo médio de 0,98 segundos por execução, operando em faixa de tempo próxima à da Busca Tabu, que exigiu 0,96 segundos, mas entregando maior aproximação à função objetivo ideal.

A separação dos resultados por tipologia de instância explicita o comportamento do VND nos diferentes conjuntos. O gap residual médio final do VND registrou 1,62% no Conjunto A, 1,14% no Conjunto B e 2,38% no Conjunto X, apresentando estabilidade na melhoria das rotas dentro do tempo limite de execução. Complementarmente à qualidade das rotas, o VND convergiu para a BKS em 17 das 50 instâncias avaliadas (34,00% do conjunto total). Essa convergência dividiu-se em 11 instâncias do Conjunto A (55,00% do subconjunto), 4 instâncias do Conjunto B (26,67% do subconjunto) e 2 instâncias do Conjunto X (13,33% do subconjunto).

A comparação dos cenários em que o VND alcançou a BKS com aqueles em que não atingiu o ótimo indica que o refinamento local é fortemente condicionado pela escala combinatória e pela densidade da instância. Nas instâncias onde a BKS foi atingida, a média de pedidos é de 3.934 distribuídos em 171 corredores ($R = 19,62$), com densidade média de 29,8 itens por corredor. Por outro lado, nas instâncias em que o VND não atingiu o BKS, a média salta para 17.397 pedidos e 333 corredores ($R = 42,05$), com densidade de 68,2 itens por corredor, atingindo em média 61,02 pedidos e 91,4 itens por prateleira no Conjunto X. Nas instâncias mais

densas, as decisões iniciais de abertura de corredores da heurística gulosa podem travar a rota em ótimos locais estruturais. Como os operadores do VND atuam por exploração de vizinhanças sem mecanismos de memória global, reestruturar o conjunto de corredores ativados exigiria fechar e reabrir simultaneamente múltiplos corredores interdependentes, realizando um salto no espaço de busca que excede a capacidade de movimentos locais isolados e justifica a necessidade das metaheurísticas subsequentes.

5.3.4 Síntese e Comparação Geral das Estruturas de Busca Local e Metaheurísticas Clássicas

A análise agregada dos dados indica que a estrutura de vizinhanças do VND proporcionou os maiores incrementos na função objetivo nesta etapa do algoritmo. O método alcançou o maior ganho absoluto em 92% das instâncias avaliadas, apresentando também a melhor relação entre ganho e tempo computacional entre as estratégias de busca local testadas.

Os métodos baseados em vizinhança única, como o LAHC com tempo médio de 0,08 segundos, apresentaram menor tempo de processamento, mas tiveram dificuldade para contornar ótimos locais em instâncias com maior densidade de pedidos por corredor. Estratégias que utilizam memória de curto prazo, como a Busca Tabu, apresentaram maior capacidade de exploração, obtendo ganho médio de 6,05 pontos na função objetivo com tempo médio de 0,95 segundos por execução.

O VND supera as abordagens de vizinhança única ao combinar múltiplos operadores complementares. Com tempo médio de execução de 0,98 segundos, comparável ao tempo da Busca Tabu, o VND alcançou a BKS em 34,00% das instâncias avaliadas. Com base nesses resultados, o VND foi selecionado como o módulo de busca local intensificadora para compor as metaheurísticas GRASP Reativo e AR-BPSO, respeitando o limite de 600 segundos por instância.

5.4 Desempenho e Comparação das Metaheurísticas Reativas

Nesta seção, avalia-se o desempenho do *framework* algorítmico proposto integrando as estratégias de construção reativa, o enxame de partículas adaptativo e a busca local intensificadora. A análise examina o comportamento das metaheurísticas GRASP Reativo e AR-BPSO nas 50 instâncias dos Conjuntos A, B e X, avaliando a qualidade das soluções finais e o tempo de convergência resultantes da atuação integrada dos mecanismos de memória adaptativa, transições de estado e paralelismo computacional.

A Tabela 5.3 apresenta a consolidação dos resultados entre as metaheurísticas GRASP Reativo e AR-BPSO propostas neste trabalho e as abordagens de referência da literatura. A coluna Inst. identifica a instância avaliada; a coluna Ped. informa o número total de pedidos; a coluna Cor. indica a quantidade de corredores disponíveis; a coluna Itens Totais descreve o

somatório de itens demandados; e a coluna BKS reporta o valor da melhor solução conhecida para a instância, disponibilizada como limite superior. As colunas Lit. 1 e Lit. 2 apresentam os desvios percentuais reportados na literatura por Santos e Baldotto (2025) e Saraiva *et al.* (2025), respectivamente. Para as duas metaheurísticas avaliadas, GRASP e BPSO, a coluna Best reporta o valor absoluto da melhor solução encontrada nas 10 execuções; a coluna Avg apresenta a média aritmética das funções objetivo obtidas; a coluna T(s) indica o tempo computacional médio em segundos demandado para alcançar a melhor solução; e a coluna Gap exibe o desvio percentual da melhor solução em relação à BKS, calculado de acordo com a Equação 5.1. Os gaps marcados em negrito indicam que os métodos alcançaram o BKS reportado pela competição ($0,00\%$). Os gaps sublinhados indicam cenários nos quais os algoritmos propostos registraram desvios percentuais menores do que os resultados publicados nos trabalhos de referência (Lit. 1 e Lit. 2), mesmo sem convergir para o BKS.

Tabela 5.3 – Comparativo de resultados entre as heurísticas GRASP e BPSO

Inst.	Ped.	Cor.	Itens Totais	BKS	Lit. 1	Lit. 2	GRASP				BPSO			
							Best	Avg	T(s)	gap	Best	Avg	T(s)	gap
A01	61	116	226	15,00	0%	0,00%	15,00	15,00	0,44	**0,00%**	15,00	15,00	0,87	**0,00%**
A02	7	33	11	2,00	0%	0,00%	2,00	2,00	0,01	**0,00%**	2,00	2,00	0,02	**0,00%**
A03	82	124	344	12,00	0%	0,00%	12,00	12,00	0,90	**0,00%**	12,00	12,00	2,54	**0,00%**
A04	16	91	71	3,50	0%	0,00%	3,50	3,50	0,06	**0,00%**	3,50	3,50	0,18	**0,00%**
A05	2625	161	13813	177,88	0%	0,56%	177,88	177,62	70,75	**0,00%**	177,88	177,88	107,15	**0,00%**
A06	10341	184	14732	691,00	0%	1,59%	691,00	691,00	56,79	**0,00%**	691,00	691,00	82,39	**0,00%**
A07	8320	180	10885	392,25	0%	0,83%	392,25	391,12	100,42	**0,00%**	392,25	392,25	190,86	**0,00%**
A08	2185	168	13034	162,94	0%	1,80%	162,82	162,14	43,16	0,07%	162,94	162,83	131,69	**0,00%**
A09	70	304	279	4,42	0%	0,00%	4,42	4,33	2,31	**0,00%**	4,42	4,35	6,22	**0,00%**
A10	1602	383	4645	17,11	1%	4,71%	15,76	15,47	23,61	7,89%	17,00	16,93	186,98	0,64%
A11	1029	375	3632	16,85	0%	11,75%	16,18	16,05	20,35	3,98%	16,18	16,09	61,87	3,98%
A12	133	342	525	11,25	0%	0,00%	11,25	11,25	2,27	**0,00%**	11,25	11,25	9,08	**0,00%**
A13	8375	413	9905	117,38	0%	13,10%	117,23	117,23	83,10	0,13%	116,85	115,58	252,93	0,45%
A14	12402	413	14980	181,64	0%	2,00%	181,45	181,42	151,14	0,10%	181,36	181,08	493,43	0,15%
A15	7367	402	9008	149,33	0%	2,90%	149,33	149,33	47,38	**0,00%**	149,33	149,30	126,72	**0,00%**
A16	1108	88	1108	85,00	0%	0,00%	85,00	85,00	3,27	**0,00%**	85,00	85,00	7,33	**0,00%**
A17	417	83	417	36,50	0%	0,00%	36,50	36,50	1,40	**0,00%**	36,50	36,50	3,41	**0,00%**
A18	2682	90	3105	117,20	0%	0,00%	117,20	117,16	13,46	**0,00%**	117,20	117,20	23,97	**0,00%**
A19	2257	134	2772	202,00	0%	0,00%	202,00	202,00	8,91	**0,00%**	202,00	202,00	16,44	**0,00%**
A20	5	5	15	5,00	0%	0,00%	5,00	5,00	0,01	**0,00%**	5,00	5,00	0,01	**0,00%**
		Médias			0,05%	1,96%			31,49	0,61%			85,20	0,26%
B01	11321	185	14973	631,50	0%	-	631,50	631,50	134,24	**0,00%**	631,50	631,50	185,36	**0,00%**
B02	2162	165	14365	180,64	0%	-	180,64	180,28	26,52	**0,00%**	180,64	180,64	62,57	**0,00%**
B03	2070	164	13013	149,60	0%	-	149,60	149,60	20,18	**0,00%**	149,60	149,60	53,51	**0,00%**
B04	2448	163	14195	155,94	0%	-	155,67	155,29	36,34	0,17%	155,67	155,27	119,12	0,17%
B05	6468	179	9489	368,00	0%	-	368,00	367,97	49,20	**0,00%**	368,00	368,00	94,34	**0,00%**
B06	1857	165	9380	84,03	0%	-	82,19	81,51	28,95	2,19%	83,80	83,45	88,19	0,27%
B07	1841	165	2331	108,00	0%	-	108,00	108,00	4,04	**0,00%**	108,00	108,00	11,11	**0,00%**
B08	12334	398	14979	227,10	0%	-	226,00	225,31	150,93	0,48%	226,80	225,95	579,43	0,13%
B09	5581	417	7320	82,50	0%	-	82,43	82,26	36,41	0,08%	82,29	82,00	147,32	0,25%
B10	14952	482	18870	444,00	0%	-	443,33	439,23	151,60	0,15%	442,00	441,13	601,50	0,45%
B11	45112	482	53747	898,80	2%	-	887,10	877,98	181,26	1,30%	858,00	806,09	608,19	4,54%
B12	28263	483	31451	567,64	0%	-	562,27	557,76	153,83	0,95%	555,36	517,10	605,93	2,16%

Tabela 5.3 – Continuação

Inst.	Ped.	Cor.	Itens Totais	BKS	Lit. 1	Lit. 2	GRASP				BPSO			
							Best	Avg	T(s)	gap	Best	Avg	T(s)	gap
B13	38214	483	49582	757,00	0%	-	749,00	740,79	158,97	1,06%	738,11	708,63	607,51	2,50%
B14	38202	483	49152	885,17	3%	-	858,33	848,65	159,30	3,03%	880,50	873,40	606,71	0,53%
B15	11541	398	14981	166,14	0%	-	161,27	160,44	147,04	2,93%	161,20	159,47	486,48	2,97%
Médias					0,33%	-			95,92	0,82%			323,82	0,93%
X01	1949	164	7362	70,85	-	-	68,53	68,43	29,67	3,28%	70,84	70,84	125,23	0,01%
X02	2942	168	3992	217,67	-	-	217,67	217,67	17,72	**0,00%**	217,67	217,67	38,99	**0,00%**
X03	11330	180	14964	667,67	-	-	667,67	667,43	150,84	**0,00%**	667,67	667,57	350,93	**0,00%**
X04	2732	172	14471	202,75	-	-	197,33	196,07	79,30	2,67%	202,50	202,15	176,41	0,12%
X05	11024	182	14911	628,67	-	-	628,67	628,67	144,22	**0,00%**	628,67	628,67	232,00	**0,00%**
X06	2704	168	14105	191,86	-	-	191,57	189,64	78,20	0,15%	191,86	191,86	114,39	**0,00%**
X07	8725	418	11079	115,15	-	-	112,79	112,67	83,47	2,06%	114,92	114,24	311,71	0,20%
X08	20364	482	24761	438,22	-	-	420,30	418,51	153,41	4,09%	408,70	404,45	603,06	6,74%
X09	39701	483	50179	819,88	-	-	799,88	789,55	160,85	2,44%	802,62	793,16	606,64	2,10%
X10	44193	482	54944	964,62	-	-	916,44	907,66	167,46	4,99%	880,67	875,26	607,33	8,70%
X11	38657	483	47957	1060,00	-	-	1056,50	1036,75	156,77	0,33%	1056,50	1055,15	605,56	0,33%
X12	52322	483	61503	1339,67	-	-	1326,67	1313,27	165,12	0,97%	1332,67	1331,30	607,91	0,52%
X13	49992	483	57285	1060,71	-	-	1034,14	1013,20	185,34	2,51%	1050,86	1044,79	610,16	0,93%
X14	68064	483	79715	1633,67	-	-	1604,67	1578,67	172,43	1,78%	1621,00	1618,43	612,90	0,78%
X15	2839	171	14688	263,91	-	-	262,91	262,51	53,49	0,38%	263,64	263,18	213,81	0,10%
Médias					-	-			119,89	1,71%			387,80	1,37%
Médias Globais					0,17%	1,96%			77,34	1,00%			247,57	0,79%

5.4.1 Análise do GRASP Reativo (R-GRASP)

A metaheurística GRASP Reativo utiliza a construção gulosa aleatorizada com um mecanismo de aprendizado em linha baseado na média móvel exponencial, identificada pela sigla EMA. Ao longo das iterações, o algoritmo atualiza dinamicamente as probabilidades de seleção entre as estratégias construtivas orientadas a pedido e orientadas a corredor, bem como os valores do parâmetro de aleatoriedade α da lista restrita de candidatos. A parametrização do expoente de reatividade, definido empiricamente como 6 para intensificar a diferenciação entre estratégias bem-sucedidas e subótimas, assim como o critério de parada, foram definidos com base em testes preliminares de calibração para respeitar o limite temporal de 600 segundos.

Em um total de 50 instâncias avaliadas com 10 repetições independentes cada, totalizando 500 execuções, o GRASP Reativo obteve a melhor solução conhecida (BKS) em 23 instâncias, o equivalente a 46,00% do conjunto total. Em 16 dessas instâncias, todas as 10 repetições convergiram para o valor da BKS com desvio padrão nulo. O desvio percentual médio da melhor solução, calculado segundo a Equação 5.1, situou-se em 1,00%. O tempo computacional médio foi de 77,34 segundos por instância.

A análise por tipologia de instância detalha o comportamento da abordagem construtiva reativa:

- No Conjunto A, composto por 20 instâncias de porte pequeno e médio, o GRASP Reativo alcançou a BKS em 15 instâncias, correspondendo a 75,00% do subconjunto. O desvio médio da melhor solução foi de 0,61% e o tempo médio de execução registrou 31,49 segundos. Na instância A10, caracterizada por restrição de estoque, o método obteve seu maior desvio percentual no conjunto, registrando 7,89%. Esse comportamento decorre da ausência de memória global entre construções independentes para transpor gargalos de alocação de itens.

- No Conjunto B, constituído por 15 instâncias de grande escala computacional, o algoritmo apresentou desvio médio de 0,82% da melhor solução e tempo médio de 95,92 segundos, alcançando a BKS nas instâncias B01, B02, B03, B05 e B07. Na instância B11, que possui 45.112 pedidos e 53.747 itens demandados, o GRASP Reativo obteve desvio de 1,30% em 181,26 segundos, superando a literatura exata que reportou gap de 2,00% nesta mesma instância. Como o tempo computacional gasto em cada construção inicial é reduzido, o algoritmo consegue gerar e refinar um expressivo volume de rotas alternativas com o VND, mantendo a exploração contínua do espaço combinatório dentro do tempo limite.

- No Conjunto X, formado por 15 instâncias com densidade de itens por corredor, o GRASP Reativo registrou desvio médio de 1,71% na melhor solução e tempo médio de 119,89 segundos. O algoritmo convergiu para a BKS nas instâncias X02, X03 e X05, apresentando desvio máximo de 4,99% na instância X10.

5.4.2 Análise do AR-BPSO

O algoritmo AR-BPSO emprega uma população de partículas operando em paralelismo por meio da biblioteca `rayon` em Rust. A dimensão da população, fixada em 100 partículas, assim como os coeficientes adaptativos de inércia e velocidade, foram calibrados empiricamente em testes preliminares para equilibrar exploração e intensificação no limite de 600 segundos. Para evitar estagnação prematura em mínimos locais, o método incorpora um mecanismo de transição de estados que alterna entre convergência, diversificação e turbulência, modificando a influência da melhor posição individual e da melhor posição global da população.

Ao longo das 50 instâncias avaliadas, o AR-BPSO alcançou a BKS em 25 instâncias, o que representa 50,00% do conjunto avaliado. O desvio percentual médio da melhor solução foi de 0,79% e o tempo computacional médio foi de 247,57 segundos.

O comportamento do AR-BPSO nos subconjuntos avaliados aponta o impacto da memória coletiva:

- No Conjunto A, o AR-BPSO atingiu a BKS em 16 das 20 instâncias, totalizando 80,00% do subconjunto, com desvio médio de 0,26% e tempo médio de 85,20 segundos. Na instância

A10, o algoritmo reduziu o desvio residual para 0,64%, superando o gap de 1,00% reportado pela literatura exata e o gap de 4,71% reportado pela literatura heurística.

- No Conjunto B, o enxame obteve desvio médio de 0,93% e tempo médio de 323,82 segundos, alcançando a BKS nas instâncias B01, B02, B03, B05 e B07. O paralelismo entre as 100 partículas viabilizou a exploração das instâncias de maior densidade dentro do tempo limite. Nas instâncias B06 e B14, o método registrou desvios de 0,27% e 0,53%, respectivamente, sendo que em B14 este resultado supera o gap de 3,00% reportado pelo método exato na literatura.

- No Conjunto X, o AR-BPSO alcançou a BKS em 4 instâncias, correspondendo a X02, X03, X05 e X06, com desvio médio geral de 1,37% no subconjunto e desvios inferiores a 0,60% nas instâncias X01, X04, X07, X11 e X12.

5.4.3 Análise Comparativa: R-GRASP vs. AR-BPSO

Os dados consolidados na Tabela 5.3 apontam diferenças de comportamento entre a construção de trajetórias independentes do GRASP Reativo e a evolução populacional do AR-BPSO.

O AR-BPSO obteve menor desvio percentual médio em relação à BKS considerando o conjunto completo de 50 instâncias, registrando 0,79% contra 1,00% do GRASP Reativo, além de convergir para a BKS em 25 instâncias contra 23 do GRASP. O compartilhamento de informação global entre as partículas contribuiu para o desempenho em instâncias com maior interdependência na ativação de corredores, como observado em A10, A13, B06, B14, X01 e X11.

A análise segmentada por subconjuntos detalha o desempenho de cada abordagem. No Conjunto A, o AR-BPSO obteve o menor desvio médio, com 0,26%, e demandou 85,20 segundos. Em comparação, o GRASP Reativo registrou desvio médio de 0,61% com tempo de 31,49 segundos. No Conjunto X, o AR-BPSO apresentou menor desvio médio residual, atingindo 1,37% em 387,80 segundos, ante os 1,71% obtidos pelo GRASP em 119,89 segundos. No Conjunto B, entretanto, o GRASP Reativo alcançou menor desvio e menor tempo de execução, com gap médio de 0,82% em 95,92 segundos, enquanto o AR-BPSO registrou 0,93% demandando 323,82 segundos. O GRASP obteve desvios menores nas instâncias com maior volume de pedidos. Na instância B11, por exemplo, o GRASP registrou desvio de 1,30% frente a 4,54% do AR-BPSO, e na instância B12 alcançou 0,95% de desvio, enquanto o BPSO obteve 2,16%.

No panorama geral, o tempo computacional médio do GRASP Reativo foi consideravelmente menor (77,34 segundos), enquanto o AR-BPSO demandou em média 247,57 segundos. A compilação nativa em Rust e o paralelismo via biblioteca `rayon` viabilizaram a execução de ambos os métodos no tempo limite de 600 segundos estipulado no protocolo experimental.

5.5 Comparativo com o Estado da Arte (Literatura)

A comparação entre os resultados obtidos pelos algoritmos propostos e as abordagens publicadas na literatura permite contextualizar o desempenho dos métodos no Problema da Seleção de Pedidos Ótima (PSPO).

5.5.1 Comparação com Métodos Exatos e Heurísticos

A Tabela 5.3 permite confrontar os resultados do *framework* proposto com duas abordagens publicadas na literatura para o mesmo problema logístico.

Na pesquisa conduzida por Saraiva *et al.* (2025), os autores reportam o desempenho tanto de uma heurística *Generate-and-Solve*, identificada pela coluna Lit. 2, quanto da resolução exata do modelo de Programação Não-Linear Inteira (PNLI) por meio do solver comercial Gurobi. A análise detalhada desses dados indica que, ao tentar resolver as instâncias médias e grandes de forma exata, o solver encerra o tempo limite de 600 segundos registrando desvios duais elevados, acima de 80%. Esses percentuais não refletem a incapacidade de encontrar boas soluções viáveis, mas sim a distância para o limitante superior dual relaxado da árvore de busca. Esse comportamento reflete a complexidade computacional de atestar a otimalidade no PSPO por vias exatas em tempo restrito, o que corrobora a necessidade de métodos heurísticos.

Ao comparar o desempenho entre as abordagens heurísticas, constata-se que o R-GRASP e o AR-BPSO propostos neste trabalho competem de forma equivalente ou superior com a heurística *Generate-and-Solve* de Saraiva *et al.* (2025) na obtenção das melhores soluções viáveis, superando os resultados de referência em instâncias desafiadoras como A05, A06, A07, A08, A10, A11, A13, A14 e A15, além de atingir desvio nulo em todas as instâncias onde a heurística de referência também convergiu para o ótimo.

A reformulação linear exata, identificada pela coluna Lit. 1 e desenvolvida por Santos e Baldotto (2025), atinge a otimalidade com desvio nulo em 27 das 35 instâncias avaliadas pelos autores dentro de 600 segundos. Entretanto, em cenários de maior densidade combinatória e complexidade operacional, notadamente nas instâncias A10, B11 e B14, a árvore de busca atinge o tempo limite de execução sem fechar o gap dual, reportando soluções viáveis com desvios residuais de 1,00%, 2,00% e 3,00%, respectivamente. Adicionalmente, as formulações exatas da literatura não relatam resultados para as instâncias do Conjunto X, que concentram os patamares mais elevados de sobreposição e demanda por corredores.

5.5.2 Novos Marcos Estabelecidos

A comparação com os trabalhos da literatura indica que o *framework* desenvolvido propicia alcançar soluções com desvios residuais menores em relação à BKS quando confrontado tanto com a abordagem heurística (Lit. 2) quanto com as limitações de tempo da formulação

exata (Lit. 1).

Em relação à heurística *Generate-and-Solve* (Saraiva *et al.* (2025), coluna Lit. 2), os métodos propostos encontraram, em suas melhores execuções, valores de função objetivo superiores em 9 instâncias que apresentavam gap em relação ao BKS. Nas instâncias A05, A06, A07, A08 e A15, onde a heurística de referência encerrou com gaps entre 0,56% e 2,90% em seus melhores testes, tanto o GRASP Reativo quanto o AR-BPSO fecharam o gap e alcançaram a melhor solução conhecida ($0,00\%$) em suas melhores execuções. Em instâncias nas quais a heurística de referência registrou desvios mais elevados, as melhores execuções dos métodos propostos apresentaram gaps reduzidos. Na instância A11, onde a Lit. 2 reportou gap de 11,75%, ambos os algoritmos alcançaram 3,98%. Na instância A13, ante um gap de 13,10% na literatura, o GRASP Reativo atingiu 0,13% e o AR-BPSO, 0,45%. Na instância A14, onde o gap era de 2,00%, obteve-se 0,10% com o GRASP e o 0,15% com o BPSO. Na instância A10, em que a heurística registrou 4,71%, o AR-BPSO convergiu para 0,64%. Cabe observar que, por se tratarem de métodos estocásticos, a função objetivo de uma execução individual pode oscilar em torno da média reportada (coluna *Avg*).

Em relação ao modelo de Programação Linear Inteira Mista (Santos e Baldotto (2025), coluna Lit. 1), os algoritmos propostos encontraram soluções viáveis com desvios menores em três instâncias complexas nas quais o método exato atingiu o limite de 600 segundos sem fechar o gap dual. Na instância A10 (1.602 pedidos, 383 corredores e 4.645 itens), a formulação exata encerrou com gap de 1,00%. O AR-BPSO obteve solução com desvio de 0,64%, auxiliado pelo mecanismo adaptativo de transição de estados. Na instância B11 (45.112 pedidos, 482 corredores e 53.747 itens), onde o método exato reportou gap de 2,00%, o GRASP Reativo obteve solução com desvio de 1,3017% em 181,26 segundos. Por fim, na instância B14 (38.202 pedidos, 483 corredores e 49.152 itens), perante o gap de 3,00% reportado pelo modelo exato, o AR-BPSO alcançou solução viável com desvio de 0,53% no tempo limite.

5.6 Evolução Incremental do Pipeline Algorítmico

A avaliação das três etapas modulares que compõem o *framework* proposto permite quantificar a progressão da qualidade das soluções e examinar o compromisso entre ganho de função objetivo e custo computacional.

A Tabela 5.4 apresenta a consolidação dos indicadores de desempenho e custo computacional ao longo das etapas metodológicas do pipeline algorítmico proposto.

A primeira coluna, intitulada Indicador / Métrica, especifica as sete métricas quantitativas avaliadas ao longo do estudo: a linha Gap Conjunto A reporta o desvio percentual médio alcançado nas 20 instâncias de porte pequeno e intermediário; a linha Gap Conjunto B exibe o desvio percentual médio nas 15 instâncias de grande escala do desafio; a linha Gap Conjunto X indica o desvio percentual médio nas 15 instâncias com alta sobreposição e densidade de itens por

corredor; a linha Gap Média Geral apresenta a média aritmética consolidada do desvio percentual considerando todo o conjunto experimental de 50 instâncias; a linha Redução de Gap expressa o ganho de qualidade da respectiva etapa em relação ao estágio anterior do pipeline, quantificado em pontos percentuais (p.p.)[1]; a linha BKS Atingidas indica a porcentagem de instâncias em que a respectiva etapa encontrou o valor da melhor solução conhecida; e a linha Tempo Médio T(s) exibe o tempo computacional médio de processamento por instância em segundos.

As quatro colunas seguintes apresentam os dados consolidados para cada estágio ou fase do pipeline algorítmico: a coluna Fase 1: AC-A reporta o desempenho obtido exclusivamente pela heurística construtiva adaptativa orientada por corredores, servindo como ponto de partida da otimização; a coluna Fase 2: + VND consolida o menor desvio atingido após submeter as abordagens construtivas à busca local multi-vizinhança; a coluna Fase 3: R-GRASP descreve os indicadores consolidados pela metaheurística GRASP Reativo completa; e a coluna Fase 3: AR-BPSO exibe os resultados da otimização populacional via enxame de partículas adaptativo.

Tabela 5.4 – Evolução comparativa dos indicadores de desvio, qualidade e tempo computacional por fase do pipeline

Indicador / Métrica	**Fase 1: AC-A**	**Fase 2: + VND**	**Fase 3: R-GRASP**	**Fase 3: AR-BPSO**
Gap Conjunto A	14,44%	1,62%	0,61%	0,26%
Gap Conjunto B	2,79%	1,14%	0,82%	0,93%
Gap Conjunto X	3,44%	2,38%	1,71%	1,37%
Gap Média Geral	7,64%	1,71%	1,00%	0,79%
Redução de Gap	–	+5,94 p.p.	+0,70 p.p.	+0,91 p.p.
BKS Atingidas	24,00%	34,00%	46,00%	50,00%
Tempo Médio T(s)	$< 0{,}05$	0,98	77,34	247,57

A análise da progressão dos dados apresentados na Tabela 5.4 ilustra a contribuição de cada módulo na transição entre rapidez de execução e refinamento global.

5.7 Síntese e Considerações Finais dos Resultados

Os experimentos computacionais conduzidos neste capítulo proporcionaram uma avaliação quantitativa do *framework* algorítmico proposto para o Problema da Seleção de Pedidos Ótima (PSPO), cobrindo as 50 instâncias logísticas reais do desafio no LVII Simpósio Brasileiro de Pesquisa Operacional sob o limite fixo de 600 segundos.

A consolidação das evidências empíricas ao longo das seis etapas analíticas fundamenta a arquitetura modular da solução.

Na primeira etapa construtiva, constatou-se que a superioridade comparativa entre as abordagens é delimitada pela razão combinatória de densidade $R = \frac{\text{Ped.}}{\text{Cor.}}$. Em cenários operacionais

[1] A utilização de pontos percentuais (p.p.), em detrimento do símbolo de porcentagem (%), faz-se necessária para evitar ambiguidades matemáticas. A métrica reporta a diferença aritmética absoluta entre dois desvios que já são intrinsecamente valores percentuais, e não uma redução relativa.

de baixa concorrência ($R < 1$), as estratégias orientadas a pedidos (OC/OC-A) alcançam desvios residuais menores ao evitar a ativação desnecessária de prateleiras. Em contrapartida, como a expressiva maioria das instâncias opera no regime de alta concorrência por prateleiras ($R \geq 1$), a consolidação orientada por corredores acoplada ao recálculo adaptativo guloso (AC-A) supera no cômputo global as estratégias orientadas a pedido, aglutinando coletas comuns e estabelecendo o ponto de partida do pipeline com desvio médio geral de 7,64% em relação à melhor solução conhecida em tempo inferior a 0,05 segundos.

Na etapa de busca local, a exploração multi-vizinhança do VND demonstrou maior capacidade de refinamento em relação aos métodos de vizinhança única, reduzindo o desvio residual para 1,71% ao custo de 0,98 segundos médios. A complementaridade entre movimentos de pedidos e de corredores mitiga o aprisionamento em ótimos locais na sequência de atendimento.

Na avaliação das metaheurísticas reativas, verificou-se que o GRASP Reativo e o AR-BPSO apresentam desempenho consistente em instâncias de diferentes escalas e densidades. O GRASP Reativo atinge a melhor solução conhecida em 46,00% das instâncias e desvio médio global de 1,00% com tempo médio de 77,34 segundos. O AR-BPSO obtém o menor desvio médio global do estudo, registrando 0,79% e convergindo para a melhor solução conhecida em 50,00% das instâncias com tempo médio de 247,57 segundos.

Na comparação com os trabalhos de referência, os algoritmos propostos reduziram o erro residual documentado pelas abordagens Lit. 1 e Lit. 2 em relação ao limite superior do desafio (BKS). Os métodos encontraram funções objetivo com menor desvio frente ao BKS do que a heurística *Generate-and-Solve* em 9 instâncias, e apresentaram desvios residuais inferiores aos do modelo exato nas instâncias A10, B11 e B14.

6 Considerações Finais

Este trabalho investigou e propôs uma metodologia de resolução para o Problema da Seleção de Pedidos Ótima (PSPO), um desafio logístico crítico no contexto de centros de distribuição de comércio eletrônico (*e-commerce*), fundamentado nas instâncias reais e no modelo matemático proposto no LVII Simpósio Brasileiro de Pesquisa Operacional (SBPO) em parceria com o Mercado Livre (Mercado Livre, 2025). A problemática central concentrou-se na maximização da produtividade operacional de coleta em armazéns, expressa pela função objetivo fracionária que mede a razão entre a quantidade total de itens selecionados e o número de corredores logísticos ativados em uma onda de coleta (*wave*), sujeita a restrições de capacidade de processamento e disponibilidade de estoque fracionado.

Diante da complexidade combinatória *NP-Hard* e da não linearidade inerente à programação fracionária, fatores que inviabilizam a aplicação de métodos exatos para instâncias de grande escala em tempo operacional compatível, foi concebida e implementada a arquitetura algorítmica modular denominada *Optimal Order Selection Problem Metaheuristic Solver Framework* (*OOSP-MetaSolver Framework*). Desenvolvida na linguagem de programação Rust para favorecer a segurança de memória e o desempenho de código nativo, a solução utilizou representações binárias compactas via *bitsets*, avaliação incremental de estoques por meio do módulo *StockBalanceEvaluator* em complexidade $O(k)$ e paralelismo em nível de dados com a biblioteca *rayon*.

Neste capítulo conclusivo, apresenta-se a síntese dos principais achados empíricos e teóricos da pesquisa, delimitam-se as limitações estruturais do trabalho e indicam-se diretrizes promissoras para investigações futuras.

6.1 Síntese dos Principais Resultados

A condução dos experimentos computacionais abrangeu um conjunto diversificado de 50 instâncias reais divididas em três tipologias operacionais (Conjuntos A, B e X), submetidas ao limite temporal de 600 segundos estipulado pelo protocolo oficial do desafio SBPO 2025. A análise progressiva das etapas do *pipeline* algorítmico corroborou a eficácia da arquitetura modular proposta e revelou propriedades do problema.

Inicialmente, a avaliação das estratégias de construção indicou que a eficácia da heurística é condicionada pela razão de densidade entre pedidos e corredores (R). Em cenários de concorrência por prateleiras ($R \geq 1$), comuns a 88% das instâncias testadas, priorizar a consolidação de itens por corredor demonstrou maior adequação. A variação adaptativa orientada por corredores (AC-A) configurou a etapa inicial da otimização, obtendo desvio percentual médio de 7,64% em

relação à melhor solução conhecida (*Best Known Solution*, BKS). Esse construtivo convergiu diretamente para a BKS em 24,00% das instâncias com custo computacional médio inferior a 0,05 segundos.

Na etapa de refinamento local, o emprego da estratégia multi-vizinhança do *Variable Neighborhood Descent* (VND), que intercala movimentos de inserção, remoção e troca em níveis de pedidos e corredores, reduziu as alocações subótimas originadas pela topologia do armazém. O VND conferiu capacidade de refinamento frente aos métodos de vizinhança única, reduzindo o desvio residual médio de 7,64% para 1,71% e elevando a convergência para a BKS a 34,00% das instâncias, com tempo médio de 0,98 segundos.

Quanto à integração das metaheurísticas, o *framework* apresentou desempenho complementar das abordagens em nichos operacionais distintos. O GRASP Reativo (R-GRASP), estruturado com calibração dinâmica e reinícios independentes, destacou-se nos cenários de amplo volume logístico (Conjunto B). No cômputo global, reduziu o desvio percentual médio para 1,00% e alcançou a BKS em 46,00% do conjunto total com tempo médio de 77,34 segundos. Em contrapartida, o *Adaptive Reactive Binary Particle Swarm Optimization* (AR-BPSO) apresentou desempenho superior nos Conjuntos A e X, registrando o menor desvio percentual médio geral da pesquisa, com 0,79%, e atingindo a BKS em 50,00% das instâncias totais. O compartilhamento de informação global auxiliou o enxame a superar platôs de estagnação, requerendo 247,57 segundos médios de execução.

No comparativo com a literatura especializada, os algoritmos propiciaram a redução do desvio residual reportado por métodos anteriores em relação ao limite matemático da competição (BKS). Os métodos atingiram convergência para a BKS em instâncias onde heurísticas construtivas de referência falharam e encontraram soluções com erro residual inferior ao documentado na literatura exata. Frente às formulações de Programação Linear Inteira Mista (Santos; Baldotto, 2025), que encerraram a execução no limite de tempo sem atestar otimalidade, o *OOSP-MetaSolver Framework* obteve soluções viáveis com desvios residuais menores nas instâncias A10, B11 e B14. Na instância B11, a abordagem registrou desvio de 1,30% em relação à BKS, contra 2,00% do modelo exato truncado. Complementarmente, o *framework* igualou ou apresentou menor erro residual que os resultados obtidos pela heurística *Generate-and-Solve* (Saraiva *et al.*, 2025) nas instâncias testadas, estabelecendo rotas de coleta competitivas e aderentes às restrições do problema.

De modo geral, os resultados indicam que o *OOSP-MetaSolver Framework* apresenta desempenho competitivo para o problema de seleção e coleta de pedidos abordado. A integração de heurísticas adaptativas com a implementação em código nativo propiciou a obtenção de soluções com desvios percentuais reduzidos frente aos métodos publicados na literatura. Dessa forma, o trabalho oferece um ferramental aplicável às operações logísticas e estabelece uma base algorítmica para futuras investigações em Pesquisa Operacional.

6.2 Limitações do Trabalho

A despeito dos resultados alcançados, é necessário contextualizar o alcance das conclusões em função das limitações inerentes à pesquisa. Primeiramente, a modelagem aborda a problemática sob uma perspectiva estática da onda de coleta, otimizando o agrupamento a partir de uma listagem consolidada do *backlog* no momento de início do algoritmo. Essa abordagem não contempla a chegada contínua e estocástica de novos pedidos durante a operação (*online order arrival*). Ademais, a modelagem adotou uma natureza mono-objetivo focada unicamente na otimização da densidade de itens por corredor visitado. Tal métrica induz um ganho indireto de produtividade global, porém omite restrições essenciais para janelas logísticas críticas, a exemplo da minimização do atraso (*tardiness*) frente a prazos de expedição e da roteirização geométrica pormenorizada dos equipamentos dentro do armazém.

Outro ponto de limitação refere-se à calibração paramétrica das metaheurísticas. Devido a restrições computacionais e de escopo, a configuração dos parâmetros nominais, tais como as dimensões populacionais, limiares iterativos e expoentes de reatividade, adveio de uma calibração empírica exploratória voltada ao cumprimento da janela de 600 segundos, não contemplando a formalidade de um *Design of Experiments* (DoE).

6.3 Trabalhos Futuros

Como desdobramento natural dos achados desta monografia e com o intuito de expandir a capacidade de resolução do *OOSP-MetaSolver Framework*, sugerem-se linhas de investigação algorítmica e metodológica para trabalhos futuros. Recomenda-se, prioritariamente, contornar a limitação imposta pela calibração empírica mediante a condução sistemática da sintonia de hiperparâmetros das metaheurísticas, valendo-se de ferramentas dedicadas à configuração automática de algoritmos, a exemplo do pacote *irace*. O ajuste científico das taxas adaptativas de aprendizagem (EMA) e do fator de inércia do enxame tende a otimizar a estabilidade das abordagens para distribuições estocásticas não previstas nos conjuntos originais.

No contexto algorítmico, a adoção de metodologias mat-heurísticas configura uma progressão metodológica promissora, almejando acoplar o desempenho de processamento paralelo e avaliação rápida do Rust com rotinas analíticas exatas. Tal modelo viabilizaria a delegação da restrição de vizinhanças para as heurísticas construtivas e o emprego circunscrito de *solvers* comerciais na extração ótima de arranjos locais. Paralelamente, sugere-se expandir as vizinhanças do refinamento local substituindo a técnica de descida pelo princípio de ruína e recriação, a exemplo da *Adaptive Large Neighborhood Search* (ALNS), favorecendo saltos combinatórios mais amplos na matriz de pedidos.

Por fim, o referencial metodológico pode ser estendido à implementação de modelos de predição baseados em aprendizado de máquina para a Seleção de Algoritmos (*Algorithm*

Selection), e à incorporação de novos paradigmas evolutivos com forte aptidão estrutural em topologias binárias, a exemplo de Algoritmos Genéticos, com o intuito de aferir o desempenho do compartilhamento genotípico em contrapartida ao direcionamento de velocidade do enxame em cenários estocásticos dinâmicos.

Referências

APPLEGATE, D. L. *et al.* **The traveling salesman problem: a computational study**. [S.l.]: Princeton university press, 2006.

BLUMOFE, R. D. *et al.* Cilk: An efficient multithreaded runtime system. In: **Proceedings of the fifth ACM SIGPLAN symposium on Principles and practice of parallel programming**. [S.l.: s.n.], 1995. p. 207–216.

BURKE, E. K.; BYKOV, Y. The late acceptance hill-climbing heuristic. **European Journal of Operational Research**, v. 258, n. 1, p. 70–78, 2017.

CORUZZOLO, A. M. *et al.* Order picking problem: A model for the joint optimisation of order batching, batch assignment sequencing, and picking routing. **Logistics**, v. 7, n. 3, p. 61, sep 2023. ISSN 2305-6290.

FEO, T. A.; RESENDE, M. G. C. Greedy randomized adaptive search procedures. **Journal of Global Optimization**, v. 6, n. 2, p. 109–133, 1995.

GENDREAU, M.; POTVIN, J.-Y. (Ed.). **Handbook of Metaheuristics**. 2nd ed.. ed. New York: Springer US, 2010. v. 146. (International Series in Operations Research Management Science, v. 146). ISBN 978-1-4419-1665-5.

GENDREAU, M.; POTVIN, J.-Y. Tabu search. In: GENDREAU, M.; POTVIN, J.-Y. (Ed.). **Handbook of Metaheuristics**. New York, NY: Springer US, 2010. p. 41–59. ISBN 978-1-4419-1665-5.

GLOVER, F.; LAGUNA, M. **Tabu Search**. Boston, MA: Kluwer Academic Publishers, 1997.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: **Proceedings of ICNN'95 - International Conference on Neural Networks**. Perth, WA, Australia: IEEE, 1995. v. 4, p. 1942–1948. ISBN 0-7803-2768-3.

KENNEDY, J.; EBERHART, R. C. A discrete binary version of the particle swarm algorithm. In: **Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics**. [S.l.: s.n.], 1997. v. 5, p. 4104–4108.

KLABNIK, S.; NICHOLS, C. **The Rust Programming Language**. [S.l.]: No Starch Press, 2023.

KOSTER, R. D.; LE-DUC, T.; ROODBERGEN, K. J. Design and control of warehouse order picking: A literature review. **European Journal of Operational Research**, v. 182, n. 2, p. 481–501, 2007.

LAND, A. H.; DOIG, A. G. An automatic method of solving discrete programming problems. **Econometrica**, JSTOR, v. 28, n. 3, p. 497–520, 1960.

LÓPEZ-IBÁÑEZ, M. *et al.* The irace package: Iterated racing for automatic algorithm configuration. **Operations Research Perspectives**, v. 3, p. 43–58, 2016. ISSN 2214-7160.

MATSAKIS, N. *et al.* **Rayon: A data parallelism library for Rust**. 2024. <https://github.com/rayon-rs/rayon>. Acessado em: 23 jun. 2024.

Mercado Livre. **Mercado Libre First Optimization Challenge**. [S.l.], 2025. Repositório oficial do desafio de otimização no SBPO 2025. Acesso em: 20 jan. 2026. Disponível em: <https://github.com/mercadolibre/challenge-sbpo-2025>.

PARDO, E. G. *et al.* Order batching problems: Taxonomy and literature review. **European Journal of Operational Research**, v. 313, n. 1, p. 1–24, 2024. ISSN 0377-2217. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0377221723001534>.

REEVES, C. R. Genetic algorithms. In: GENDREAU, M.; POTVIN, J.-Y. (Ed.). **Handbook of Metaheuristics**. New York, NY: Springer US, 2010. p. 109–139.

RESENDE, M. G. C.; RIBEIRO, C. C. Greedy randomized adaptive search procedures: Advances, hybridizations, and applications. In: GENDREAU, M.; POTVIN, J.-Y. (Ed.). **Handbook of Metaheuristics**. New York, NY: Springer US, 2010. p. 283–319. ISBN 978-1-4419-1665-5.

SANTOS, A. L. F. d.; BALDOTTO, P. F. Uma formulação linear e um algoritmo exato para o problema da seleção de pedidos ótima. In: **Anais do 57° Simpósio Brasileiro de Pesquisa Operacional (SBPO)**. Gramado, RS: [s.n.], 2025. Artigo submetido ao Desafio de Otimização SBPO 2025. Disponível em: <https://proceedings.science/p/212172?lang=pt-br>.

SARAIVA, R. D. *et al.* GENERATE-AND-SOLVE aplicada ao problema da seleção de pedidos. In: **Anais do LVII Simpósio Brasileiro de Pesquisa Operacional (SBPO 2025)**. Fortaleza, CE / Recife, PE: [s.n.], 2025. Artigo submetido ao Desafio de Otimização SBPO 2025. Disponível em: <https://proceedings.science/p/212377?lang=pt-br>.

SOUZA, M. J. F. **Inteligência Computacional para Otimização: meta-heurísticas**. Ouro Preto, Minas Gerais, 2024. Acesso em: 02 dez. 2025. Disponível em: <http://www.decom.ufop.br/prof/marcone/Disciplinas/Inteligencia%20Computacional/InteligenciaComputacional.pdf>.

YANG, P.; ZHAO, Z.; GUO, H. Order batch picking optimization under different storage scenarios for e-commerce warehouses. **Transportation Research Part E: Logistics and Transportation Review**, v. 136, p. 101897, 2020. ISSN 1366-5545.