UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

CARLOS WYLLIAM MONDO

OTIMIZAÇÃO DA ALOCAÇÃO DE PRODUTOS EM UMA ESTRUTURA DE ARMAZENAGEM *FLOW-RACK* UTILIZANDO PROGRAMAÇÃO LINEAR INTEIRA MISTA

PONTA GROSSA

2024

CARLOS WYLLIAM MONDO

OTIMIZAÇÃO DA ALOCAÇÃO DE PRODUTOS EM UMA ESTRUTURA DE ARMAZENAGEM *FLOW-RACK* UTILIZANDO PROGRAMAÇÃO LINEAR INTEIRA MISTA

Optimization of product allocation in a Flow-Rack storage structure using mixed-integer linear programming

Trabalho de Dissertação apresentado como requisito para obtenção do título de Mestre em Engenharia Elétrica do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná.

Orientador: Profª. Drª. Marcella Scoczynski Ribeiro Martins

Coorientador: Profª. Drª. Cristhiane Gonçalves

PONTA GROSSA

2024

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Ponta Grossa

CARLOS WYLLIAM MONDO

OTIMIZAÇÃO DA ALOCAÇÃO DE PRODUTOS EM UMA ESTRUTURA DE ARMAZENAGEM FLOW-RACK UTILIZANDO PROGRAMAÇÃO LINEAR INTEIRA MISTA

Trabalho de pesquisa de mestrado apresentado como requisito para obtenção do título de Mestre Em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná (UTFPR). Área de concentração: Controle E Processamento De Energia.

Data de aprovação: 21 de Maio de 2024

Dra. Marcella Scoczynski Ribeiro Martins, Doutorado - Universidade Tecnológica Federal do Paraná

Dr. Angelo Marcelo Tusset, Doutorado - Universidade Tecnológica Federal do Paraná

Dra. Cristhiane Goncalves, Doutorado - Universidade Tecnológica Federal do Paraná

Dr. Guilherme Alberto Sousa Ribeiro, Doutorado - Hospital Israelita Albert Einstein

Dr. Mauricio Dos Santos Kaster, Doutorado - Universidade Tecnológica Federal do Paraná

Dedico este trabalho à minha família, pelos momentos de ausência.
Aos meus pais Tullio Mondo (*In Memoriam*) e Beatriz H. Mondo que me inspiraram e sempre me incentivaram à busca pelo conhecimento, mesmo quando era apenas no silêncio que ele se encontrava.
À minha esposa Hellen por todo o apoio, confiança e paciência que me transmitiu neste período. À minha filha Helena, a pessoa mais importante de minha vida e que me deu as forças e motivação necessária para lutar e seguir em frente.

AGRADECIMENTOS

Certamente estes parágrafos não irão atender a todas as pessoas que fizeram parte dessa importante fase de minha vida. Portanto, desde já peço desculpas àquelas que não estão presentes entre essas palavras, mas elas podem estar certas que fazem parte do meu pensamento e de minha gratidão.

Agradeço a Deus, em primeiro lugar, por tudo que ele me proporcionou ao longo de minha vida.

Agradeço a minha família pela força e confiança em todos os momentos pois acredito que sem o apoio deles seria muito difícil vencer esse desafio.

Agradeço a Universidade Tecnológica Federal do Paraná que me deu a oportunidade de cursar a graduação em engenharia eletrônica e ao mestrado em engenharia elétrica em uma instituição de qualidade e excelência.

Agradeço aos meus professores orientadores Marcella Scoczynski Ribeiro Martins e Cristhiane Gonçalves pela sabedoria com que me guiaram nesta trajetória. Aos demais professores que contribuíram direta ou indiretamente para a realização do presente trabalho e para a minha formação.

Agradeço aos meus colegas e amigos, que em alguma disciplina cruzaram minha trajetória e de certo modo colaboraram e incentivaram durante esse período.

Enfim, a todos os que por algum motivo contribuíram para a realização desta pesquisa.

Comece fazendo o que é necessário, depois o
que é possível, e de repente você estará
fazendo o impossível.
(São Francisco de Assis)

RESUMO

O processo de armazenagem para a estocagem de produtos é um elemento essencial para o setor logístico, principalmente na indústria farmacêutica. Neste setor, como nos demais, deve-se garantir a qualidade dos medicamentos para uso com qualidade assegurada. Uma das soluções utilizadas é a estrutura de armazenagem *Flow-rack*, que possui compartimentos escalonados que permitem o fluxo contínuo. Sendo assim, o processo de alocação de produtos pode ser otimizado de forma a maximizar a eficiência do fluxo, considerando restrições como capacidade de carga e requisitos de frequência de saída dos produtos. Diversos métodos de otimização podem ser aplicados. A otimização linear, considerada neste trabalho, busca encontrar a melhor solução considerando uma função objetivo linear sujeita a um conjunto de restrições lineares. Para resolver o problema de otimização algumas ferramentas vêm sendo aplicadas, como a *Mixed Integer Programming* (MIP). Assim, este trabalho aplica MIP na otimização da alocação de produtos em um *Flow-rack* considerando múltiplos objetivos e restrições de forma simultânea em uma estrutura metálica de armazenagem, observando a ergonomia e visando a melhoria de produção. Foram implementados os solucionadores *Solving Constraint Integer Programs* (SCIP) e *Google Linear Optimization Package* (GLOP) baseado em *branch-and-bound*, assumindo um problema de alocação semelhante ao da mochila, clássico na literatura e o *Constraint Programming with Satisfiability* (CP-SAT) que integra técnicas de programação de restrições com métodos de resolução de *Satisfiability* (SAT) *booleana*. Os resultados demonstraram que o solucionador SCIP e o CP-SAT alocaram os produtos atendendo todos os critérios estabelecidos, como por exemplo a capacidade da mochila, e a utilização correta todos os produtos indicados no projeto.

Palavras-chave: otimização linear; programação linear inteira mista; resolução de programas inteiros com restrições; *flow-rack*; alocação de produtos.

ABSTRACT

The storage process for product stocking is an essential element for the logistics sector, especially in the pharmaceutical industry. In this sector, as in others, the quality of medicines must be ensured for use with assured quality. One of the solutions used is the Flow-rack storage structure, which has staggered compartments that allow continuous flow. Thus, the product allocation process can be optimized to maximize flow efficiency, considering constraints such as load capacity and product output frequency requirements. Various optimization methods can be applied. Linear optimization, considered in this work, seeks to find the best solution considering a linear objective function subject to a set of linear constraints. To solve the optimization problem, some tools have been applied, such as Mixed Integer Programming (MIP). Thus, this work applies MIP in the optimization of product allocation in a Flow-rack considering multiple objectives and constraints simultaneously in a metal storage structure, observing ergonomics and aiming at production improvement. The Solving Constraint Integer Programs (SCIP) and Google Linear Optimization Package (GLOP) solvers were implemented based on branch-and-bound, assuming an allocation problem similar to the backpack, classic in the literature and the Constraint Programming with Satisfiability (CP-SAT) that integrates constraint programming techniques with boolean Satisfiability (SAT) resolution methods. The results showed that the SCIP and CP-SAT solvers allocated the products meeting all the established criteria, such as the backpack capacity, and the correct use of all the products indicated in the project.

Keywords: linear optimization; mixed integer programming; solving constraint integer programs; flow-rack; product allocation.

LISTA DE ALGORITMOS

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

Siglas

CA	*Conflict Analysis*
CP-SAT	*Constraint Programming with Satisfiability*
DORT	Distúrbios Osteo-musculares Relacionados ao Trabalho
FIFO	*First In, First Out*
GLOP	*Google Linear Optimization Package*
IoT	*Internet of Things*
MIP	*Mixed Integer Programming*
NR	Norma Regulamentadora
PLC	*Programmable Logic Controller*
RAM	*Random Access Memory*
SAT	*Satisfiability*
SCIP	*Solving Constraint Integer Programs*
VSC	*Microsoft Visual Studio Code*
WMS	*Warehouse Management System*

SUMÁRIO

1 INTRODUÇÃO

A indústria farmacêutica é uma das mais importantes para a saúde humana, sendo responsável pela fabricação e distribuição de medicamentos, vacinas, suplementos, entre outros produtos. O processo de armazenagem para a estocagem de medicamentos, pode-se dizer, é um elemento importante do nível de serviço logístico, sobretudo por que é nesse processo que deve ser garantida a estabilidade dos medicamentos para uso com qualidade assegurada. (NUNES *et al.*, 2019) afirma que para o correto armazenamento, os medicamentos devem ser estocados sob condições apropriadas, com o objetivo de manter sua integridade.

Uma das soluções utilizadas para a armazenagem é a estrutura *Flow-rack*, que consiste em um sistema de prateleiras inclinadas com roletes que atua como um mecanismo para que as caixas se desloquem naturalmente devido à gravidade, tornando mais fácil acomodá-las completamente na estrutura (ALMEIDA *et al.*, 2019). Essa solução de armazenagem se destaca pela sua eficiência na movimentação dos produtos, possibilitando uma melhor distribuição dos medicamentos, redução de perdas e aumento da segurança no manuseio dos mesmos. Além disso, a estrutura *Flow-rack* permite um controle de estoque, facilitando a gestão dos produtos, evitando desperdícios e melhorando a ergonomia no ambiente de trabalho. Outra grande vantagem relacionada ao *Flow-rack* é a gestão dos prazos de validade dos produtos, onde os produtos mais novos, e com maior durabilidade, ficam no fim da fila.

No que tange à ergonomia, com a estrutura *Flow-rack* é possível reduzir a necessidade de os funcionários realizarem movimentos repetitivos e desnecessários, pois os produtos são facilmente acessíveis e podem ser retirados ou colocados sem a necessidade de esforço excessivo ou posturas inadequadas. Em relação à otimização de espaço, as estruturas de armazenagem *Flow-rack* oferecem a vantagem de maximizar a capacidade de armazenamento em espaços reduzidos. Isso ocorre porque os produtos são armazenados em níveis múltiplos, permitindo o uso do espaço vertical. Além disso, o uso de *Flow-racks* também melhora a eficiência operacional do armazém, uma vez que os produtos são organizados de forma a permitir uma movimentação mais fluida.

Contextualizando esse problema físico, ele pode ser interpretado como um problema de otimização combinatória, mais precisamente como o problema da mochila. Este problema de otimização caracteriza-se no preenchimento de uma mochila com objetos de diferentes pesos e valores (AMARANTE, 2013). Neste trabalho o problema é resolvido através da aplicação de algoritmos de otimização linear para encontrar a melhor solução com relação à capacidade de cada mochila e à frequência com que esses produtos são selecionados.

Dentre os solucionadores para a resolução de problemas de programação linear e não linear com restrições mistas estão o SCIP e o GLOP. Eles fornecem a opção de restrição "opto-acumulativa", introduzida para modelar restrições de recursos renováveis no contexto de tarefas com múltiplos modos. No entanto, para aplicar a mencionada restrição é necessário introduzir variáveis inteiras de tempo de início para cada tarefa e modo e as variáveis binárias para a

atribuição de modo da tarefa. A concepção modular do SCIP e dos demais solucionadores baseados em MIP e a disponibilidade do código-fonte tornam-no uma plataforma ideal para pesquisas (SCHNELL; HARTL, 2017). Além disso, seu bom desempenho possibilita que pesquisas computacionais e desenvolvimento de algoritmos na área de programação matemática considerem as interações entre novos métodos e componentes de última geração existentes, o que é essencial para o avanço contínuo dos algoritmos na tecnologia de resolução de MIP (ACHTERBERG, 2009).

A ideia dos algoritmos CP-SAT é combinar a propagação de domínio processada através de restrições globais com as técnicas de Análise de Conflitos (*Conflict Analysis* (CA)) de um resolvedor SAT. Portanto, os diferentes propagadores das restrições globais geram explicações, ou seja, cláusulas compostas por literais booleanos, para suas atualizações de domínio e as inconsistências detectadas. Essas explicações são transferidas para um mecanismo de resolução SAT. O mecanismo SAT constrói um grafo de conflitos com base nas explicações dos propagadores de domínio e pode deduzir cláusulas inválidas para um modelo SAT e movimentos de retrocesso. Em geral, os algoritmos fazem ramificações nas variáveis e valores com base no número de conflitos em que os literais respectivos estiveram envolvidos (SCHNELL; HARTL, 2017).

Os gargalos observados neste trabalho dizem respeito à velocidade em que o operador consegue realizar a seleção dos produtos, isto pode ser quantificado através de um estudo de tempos e movimentos e às condições ergonômicas às quais o operador se submete durante essa atividade. Considerando que dentro de centros de distribuição como este não há somente um operador mas dezenas deles, o problema toma dimensões maiores, sendo viável o estudo de um método que através das variáveis envolvidas neste processo possa apresentar uma solução na alocação dos produtos em estruturas de armazenagem (CASTRO; RAMOS; COSTA, 2012).

A otimização do espaço em estruturas de armazenagem é essencial para empresas de armazenamento e distribuição de mercadorias. Uma organização inadequada pode levar ao desperdício de tempo e recursos, aumentar os tempos de manuseio e movimentação dos produtos e até danificá-los (ALMEIDA *et al.*, 2019). Esta dissertação propõe melhorar a alocação de produtos em estruturas de armazenagem Flow-racks utilizando técnicas de programação linear inteira mista, com foco na ergonomia e na melhoria da produção, especialmente para produtos farmacêuticos e medicamentos. A alocação de produtos farmacêuticos e os problemas de saúde relacionados à ergonomia são desafios significativos na logística e na gestão de armazéns. A consideração da ergonomia dos funcionários é crucial para garantir um ambiente de trabalho seguro e produtivo. A pesquisa pode levar a melhorias no armazenamento e na satisfação do funcionário, impactando positivamente os números de operações.

No que diz respeito à instrumentação e controle como área maior do desenvolvimento do projeto, há a possibilidade de implementação do algoritmo com o *solver* em *Programmable Logic Controller* (PLC)s diretamente na indústria, porém geralmente possuem uma memória

limitada em alguns *megabytes* e limitado poder de processamento. Isto implica na necessidade de um sistema externo onde o *solver* MIP é carregado e um módulo de rede conectando o sistema externo e o PLC (HAN; ZHOU, 2010). Em conjunto com o sistema *WMS*, realiza o gerenciamento dos itens e alocação nas posições indicadas. Aliado ao gerenciamento de itens pode-se utilizar robôs *pick-and-place* que realizam a tarefa de classificar itens de um local a outro, estes itens podem vir de esteiras transportadoras. Os objetos a serem selecionados podem ser entendidos como uma fila de clientes com alteração de prioridades de forma dinâmica de acordo com a programação realizada (MATTONE; ADDUCI; WOLF, 1998).

1.1 Objetivos

1.1.1 Objetivo geral

O presente trabalho tem por objetivo geral otimizar a alocação de produtos através de técnicas de programação linear em uma estrutura metálica de armazenagem, observando a ergonomia e visando a melhoria de produção.

1.1.2 Objetivos específicos

Para o cumprimento do objetivo geral do trabalho, são listados a seguir os seguintes objetivos específicos:

- Levantamento do estado da arte a cerca dos assuntos de otimização linear utilizando solucionadores na resolução de problemas de alocação de produtos;
- Definir e delimitar um conjunto de produtos farmacêuticos para otimização de sua alocação em estruturas de armazenagem;
- Definir os critérios de distribuição conforme volume e frequência;
- Descrever métodos de otimização linear para organização dos produtos;
- Realizar a apresentação dos resultados com base nos critérios de avaliação selecionados.

1.2 Justificativa

Os centros de distribuição na indústria farmacêutica funcionam como um pulmão para agilizar o abastecimento de produtos para os comércios locais de perfumaria e medicamentos. Dentro desses centros podem ser observados diferentes tipos de sistemas de armazenagem,

desde produtos menores separados de maneira dedicada de acordo com os pedidos que chegam bem como paletes que concentram produtos de validade estendida e maior volume.

Para os produtos menores, em geral, estes galpões possuem uma área denominada *picking* ou separação de produtos. Esta separação é realizada em muitos locais de maneira manual onde o operador, ao receber um pedido através de uma lista de separação, deve coletar os produtos nos devidos endereços, colocá-los em uma caixa e liberá-lo geralmente com o auxílio de uma esteira motorizada, para uma continuação de separação.

As dificuldades identificadas neste trabalho estão relacionadas à rapidez com que o operador é capaz de selecionar os produtos e às condições ergonômicas enfrentadas durante essa tarefa. Levando em conta que em centros de distribuição como este, não existe apenas um, mas vários operadores, a questão se torna ainda mais complexa. Portanto, é apropriado investigar um método que, considerando as variáveis deste processo, possa oferecer uma solução para a disposição dos produtos em sistemas de armazenamento.

Dessa forma, um trabalho de otimização de espaço em estruturas de armazenagem pode trazer inúmeros benefícios para uma empresa. Isto também pode contribuir para a sustentabilidade, uma vez que a redução do espaço utilizado pode resultar em menor consumo de energia e recursos naturais na construção e manutenção de novas instalações. Outro ponto é a ergonomia que é fundamental em estruturas de armazenagem, pois contribui com as condições de trabalho dos operadores. Ao projetar e construir essas estruturas com este foco, são reduzidos riscos de lesões. Investir em melhorias beneficia não apenas a saúde e segurança, mas também otimiza processos.

1.3 Estrutura do trabalho

Este trabalho está organizado da seguinte forma:

O **Capítulo 2** apresenta uma descrição do problema físico encontrado, a contextualização do problema da mochila multi-objetivo e também apresenta os métodos de otimização linear MIP e CP-SAT utilizados para a solução do problema além da revisão de literatura contendo os principais conceitos relacionados ao tema deste capítulo.

O **Capítulo 3** proporciona o estado da arte acerca do tema da presente dissertação, apontando os principais trabalhos existentes que tratam sobre o problema de múltiplas mochilas.

O **Capítulo 4** aborda como a metodologia do presente trabalho foi desenvolvida, explicitando quais os métodos utilizados e as etapas de desenvolvimento, apresentando fluxograma de dados e as etapas de elaboração do algoritmo proposto.

O **Capítulo 5** demonstra os resultados preliminares obtidos com relação à aplicação da otimização aos dados de armazenagem de produtos propostos, levantando também a discussão acerca destes resultados.

Por fim, o **Capítulo 6** apresenta quais são as considerações finais sobre o presente trabalho, além das perspectivas de trabalhos futuros.

2 REFERENCIAL TEÓRICO

No presente capítulo são abordados os conceitos de armazenagem na indústria farmacêutica além de conceitos sobre o problema da mochila, múltiplas mochilas e sobre otimização linear.

2.1 Sistemas de armazenagem na indústria farmacêutica

Os sistemas de armazenagem na indústria farmacêutica devem possuir condições de armazenamento de modo a garantir a qualidade, segurança e eficácia, não causando nenhum risco ao usuário (NUNES *et al.*, 2019). Devido à sensibilidade dos produtos farmacêuticos, é essencial que as empresas farmacêuticas adotem sistemas de armazenamento adequados que atendam aos padrões regulatórios e às boas práticas de fabricação.

2.1.1 Tipos de armazenagem

Alguns tipos de armazenagem existentes serão apresentados a seguir:

Armazenagem em temperatura ambiente: A maioria dos medicamentos pode ser armazenada em temperatura ambiente, geralmente entre 15℃ e 25℃. Para isso, são utilizadas estantes ou prateleiras em ambientes controlados, protegidos da exposição direta à luz solar e umidade. O controle adequado de estoque e o registro das condições ambientais são fundamentais para garantir a integridade dos medicamentos.

Armazenagem refrigerada: Certos medicamentos, como vacinas e produtos biológicos, requerem armazenamento em temperaturas refrigeradas (entre 2℃ e 8℃) para manter sua eficácia e evitar a deterioração. Esses produtos são mantidos em câmaras frias, que devem ser monitorados por meio de equipamentos calibrados constantemente. A medição constante da temperatura e umidade deve ser realizada utilizando sondas, termômetros ou registradores de dados, que permitem monitorar esses parâmetros em tempo real (ROSA, 2023).

Armazenagem automatizada: Para otimizar o processo de gerenciamento de estoque e aumentar a eficiência, muitas indústrias farmacêuticas têm adotado sistemas de armazenagem automatizada, como robôs e transportadores motorizados que facilitam o transporte de mercadorias em instalações de armazenagem (SISTEMAS, 2023).

Todos os tipos de armazenagem citados devem seguir o princípio da rastreabilidade. Em relação ao setor farmacêutico, o rastreamento de medicamentos pode parecer apenas um processo técnico, desenvolvido para cumprir as normas regulatórias da indústria, com o obje-

tivo de coletar e guardar informações dos produtos. No entanto, essa prática é uma garantia de segurança para o consumidor, assegurando a origem confiável e a qualidade dos medicamentos comprados (PINTO, 2016). Por isso, deve-se permitir o rastreamento dos lotes de produtos, desde a fabricação até a distribuição, facilitando a localização de qualquer problema, caso necessário.

2.1.2 Estruturas de armazenagem

A atividade do *picking* manual é um processo que envolve a separação e preparação de um pedido de um cliente. O operador deve selecionar os produtos corretamente para que o processo apresente eficiência das operações de coleta em um armazém ou centro de distribuição (ROSA, 2023). Os sistemas de armazenagem são especialmente úteis em ambientes onde as operações de *picking* são frequentes e necessitam ser realizadas com agilidade e precisão. Para que esta operação seja eficiente, são necessárias estruturas de armazenagem específicas que são citadas na sequência.

Mini porta-palete: A estrutura Mini porta-palete é um sistema de armazenagem projetado especialmente para operações em sistemas de *picking* manual, dispensando a necessidade do uso de empilhadeiras.

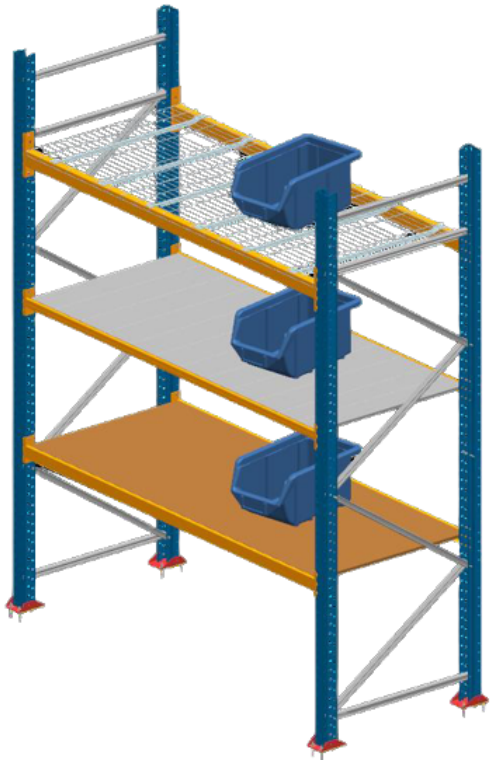
Os níveis de armazenagem da estrutura Mini porta-palete são reguláveis de acordo com as dimensões e volume das cargas e facilmente ajustáveis com um sistema de montagem e desmontagem prático e simples. Essa característica permite que a estrutura seja adaptada às necessidades da empresa, maximizando o espaço disponível para armazenamento. A estrutura Mini porta-palete possui capacidade de armazenamento para cargas leves e pesadas, utilizando planos metálicos ou em madeira que suportam o peso de até 600kg em cada plano. Esse sistema é uma excelente opção para empresas que precisam armazenar diferentes tipos de produtos em um mesmo espaço (COSTA; VALÉRIO, 2020).

Além disso, a utilização da estrutura Mini porta-palete permite uma maior organização no processo de *picking* manual, tornando a movimentação de materiais mais ágil. Com esse sistema, os produtos podem ser facilmente localizados e retirados da estrutura de armazenagem.

Pode-se observar um exemplo de Mini porta-palete, na Figura 1. A estrutura de Mini porta-palete demonstra que os planos podem ser customizáveis com a utilização de diferentes materiais como: madeira, chapas de aço e malhas de aço isto permite diversificar o armazenamento considerando tipos de produtos diferentes e aprimorar o processo de *picking* manual.

***Flow-rack*:** O *Flow-rack*, também conhecido como prateleira dinâmica ou rolo de gravidade, é um sistema de armazenamento de materiais que utiliza rolos de gravidade para movimentar os produtos ao longo da prateleira (CASTRO; RAMOS; COSTA, 2012). Essencialmente,

Figura 1 – Mini porta-palete



Fonte: Autoria própria (2024).

é uma estrutura de prateleiras inclinadas com rolos que permitem que as caixas ou produtos deslizem suavemente ao longo da prateleira até a área de retirada.

O objetivo é utilizar de maneira ótima o espaço de armazenamento e reduzir o tempo de movimentação dos produtos dentro do processo de *picking*. Este tipo de estrutura condiciona a utilização dos produtos em sequência de modo como são introduzidos, o que reduz nos desperdícios com os vencimentos dos itens administrados no estoque (ALMEIDA *et al.*, 2019).

O funcionamento do *Flow-rack* é bastante simples: os produtos são colocados no topo da prateleira e são movidos automaticamente para o fundo da mesma, à medida que outros produtos são retirados. Isso é possível graças aos rolos de gravidade, que são posicionados em um ângulo ligeiramente inclinado. Quando um produto é retirado do fundo da prateleira, os rolos de gravidade movimentam automaticamente o próximo produto para o lugar vago. Pode-se observar a organização desta estrutura na Figura 2.

Os planos para a colocação dos itens são reguláveis de acordo com as dimensões dos produtos a serem armazenados e também com a altura para a seleção pelo operador. O sistema *Flow-rack* é excelente opção para ser usado em conjunto com outras estruturas de metálicas de armazenagem a fim de proporcionar um melhor aproveitamento vertical. Segundo (ALMEIDA *et al.*, 2019) a adoção de uma prática alternativa ao tradicional empilhamento de produtos em paletes resulta na significativa redução dos desperdícios de itens com defeito.

Além disso, este sistema de armazenagem pode ser usado integrado com sistemas de *pick-to-light* ou outros sistemas automatizados para ordenamento de montagem de pedidos, as-

Figura 2 – *Flow-rack*

Fonte: (SISTEMAS, 2023).

sim como *put-to-light* para ordenamento da reposição do estoque do sistema. Para este trabalho esta estrutura foi escolhida pois é a que mais se ajusta a este tipo de separação de produtos e modelagem do problema da mochila.

2.2 Gestão dos itens

Aprimorar a gestão de itens reveste-se de grande relevância para os desempenhos da empresa, uma vez que os sistemas de armazenamento e administração de inventário desempenham um papel crucial para garantir a disponibilidade de recursos para atender às demandas emergentes. Além disso, como uma justificação adicional, é notável que um controle apropriado dos depósitos promova a eficácia tanto do processo quanto do serviço. Portanto, é importante avaliar minuciosamente os sistemas e procedimentos relacionados à gestão de armazenamento, identificar potenciais deficiências, identificar suas necessidades e explorar alternativas que possam melhorar a eficiência do controle de estoque e armazenagem. Ainda, de acordo com o autor, a gestão de armazéns sempre representou um desafio significativo para as organizações, uma vez que manter o estoque acessível e com prazos de atendimento reduzidos passou a ser um dos fatores-chave para a competitividade das empresas (ROCHA *et al.*, 2021).

Uma das principais responsabilidades da gestão de estoque consiste em supervisionar a quantidade de mercadorias mantidas em armazenamento, visando manter um equilíbrio adequado. Isso significa evitar tanto o acúmulo excessivo de produtos, o que resultaria em custos de manutenção elevados, quanto a escassez de produtos, que levaria à insatisfação dos clientes (ALMEIDA *et al.*, 2019).

2.2.1 Classificação ABC

A curva ABC é uma das ferramentas para um gerenciamento de estoque pois permite identificar os itens que merecem um tratamento dedicado na administração dos materiais (COSTA, 2017). De acordo com (VILLAR; SILVA; NOBREGA, 2008) esse método se baseia no princípio de Pareto, um conceito desenvolvido a partir da publicação de um livro no ano de 1906 por Pareto, que afirma que 80% das consequências são resultados de apenas 20% das causas. Devido a essa associação, esse método também é chamado de princípio da curva 80-20.

Essa ferramenta desempenha um papel crucial ao identificar itens que requerem atenção e tratamento adequados em sua gestão. Além disso, a metodologia da curva ABC tem sido amplamente aplicada para resolver diversos problemas empresariais comuns, como a gestão de estoques, a formulação de políticas e a definição de prioridades na programação da produção (DIAS, 2010).

A utilização da classificação ABC permite aos gestores identificar itens que necessitam de tratamento especial, tanto em termos de quantidade quanto em termos de impacto financeiro, otimizando assim a categorização dos componentes do estoque. Observa-se então a importância de investir em sistemas de informação e no processamento de dados para identificar e distinguir várias situações que exigem controles específicos de estoque, a fim de evitar um aumento descontrolado dos custos (ALMEIDA *et al.*, 2019).

A disposição dos itens em um sistema de armazenagem é de extrema importância na indústria farmacêutica. Com o propósito de aprimorar a utilização do espaço disponível e minimizar as perdas causadas pela movimentação dos colaboradores e danos aos produtos devido ao vencimento do prazo de validade dos medicamentos é importante que sejam priorizados sistemas de armazenagem *First In, First Out* (FIFO), que quer dizer, primeiro produto que entra é o primeiro produto que sai. Isto evita que problemas relacionados a produtos vencidos fiquem parados no estoque e gerem perdas de ativos (ALMEIDA *et al.*, 2019).

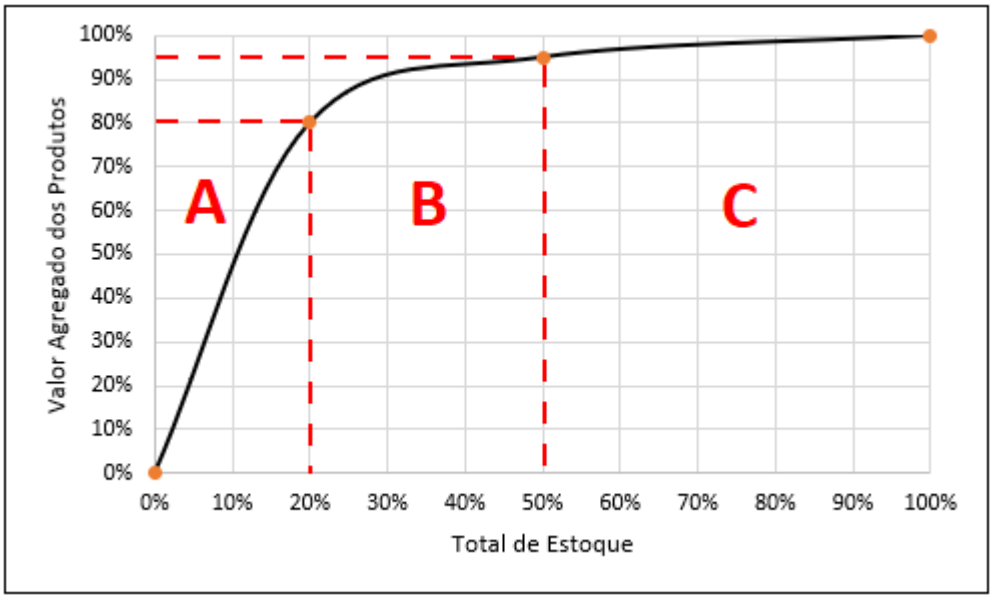
Segundo (COSTA, 2017), a maioria da atenção deve ser direcionada para os itens de maior relevância, classificados como A, os quais devem representar 80% do valor monetário dos 20% dos materiais armazenados. Os itens de classe B devem corresponder a 15% do valor monetário dos 30% dos produtos. Por fim, os itens de classe C representam 5% do valor para 50% dos produtos.

A atenção dada à quantidade de produtos em estoque aumenta devido à necessidade de uma análise minuciosa por parte do gestor de logística, a fim de garantir que a empresa possa operar com o menor investimento possível nessa área. Dado o grande número de itens armazenados pelas empresas, a aplicação da curva ABC auxilia o gestor a exercer um controle rigoroso sobre os materiais de consumo, classificados como classes A, B e C, sendo fundamental a implementação da técnica (MARTINS; LAUGENI, 2005).

Para elaborar a curva é inicialmente necessário reunir os dados dos materiais a serem categorizados. Essa etapa é considerada a mais desafiadora, pois implica na coleta de infor-

mações que englobam a identificação dos itens. Essa identificação é crucial para determinar a alocação nos depósitos ou locais de armazenamento, respeitando o limite estabelecido para a gestão. Na análise das características do produto, são considerados o peso, volume, natureza perecível, inflamabilidade e a possibilidade de substituição. Durante esse processo de coleta, é fundamental calcular os custos anuais e organizá-los devidamente (BALLOU, 2009).

Figura 3 – Exemplo de Curva ABC



Fonte: Autoria própria (2024).

Observando a Figura 3 pode-se verificar um exemplo de gráfico da curva ABC. É possível observar que os produtos classificados como A representam os artigos de maior valor agregado e representam uma parcela menor do estoque. Os produtos na categoria B ocupam uma posição intermediária e recebem uma atenção moderada, pois alguns estão próximos dos itens de maior valor agregado. Por último, os produtos classificados como C são aqueles em que o capital investido é menor porém possuem uma representatividade maior em estoque considerando a circulação das mercadorias.

Devido à variação nos padrões das curvas, (COSTA, 2017) destaca que elas se ajustam conforme a porcentagem de itens presentes na empresa, o que resulta em diferentes cenários e, por vezes, em desafios para quem está implementando o sistema. Isso ocorre porque pode haver situações em que não existem produtos classificados como A ou em que a classificação A não resultou na diminuição do estoque desses itens. No entanto, essa ferramenta serve como um auxílio para a tomada de decisões relacionadas ao estoque, identificando os itens que necessitam de redução. Com o apoio de outras ferramentas, é possível efetivar essa redução.

2.2.2 Sistema de gerenciamento

Para melhorar a distribuição dos itens, um grande aliado é o sistema *Warehouse Management System* (WMS), que se caracteriza como um *software* que auxilia na atividade de armazenamento dos itens utilizando os dados coletados para alcançar processos mais eficientes com controle e dos níveis de estoque e inventário (ROCHA *et al.*, 2021).

O sistema de WMS oferece informações abrangentes sobre o fluxo de materiais nas áreas virtuais, proporcionando uma visão ampla e sucinta da localização dos materiais dentro da empresa, das ordens de trabalho vinculadas a eles, das operações já concluídas e de quem as realizou, além de mostrar as operações pendentes. A rastreabilidade se torna consideravelmente mais eficaz, eliminando a necessidade de uso de papel. É importante destacar que, com o WMS, é possível efetuar a baixa de materiais do estoque e qualquer movimentação por item, em contraste com a situação anterior em que era necessário que todas as peças de um pedido estivessem completas para realizar a baixa (ASSIS; SAGAWA, 2018).

O WMS tem a capacidade de proporcionar ainda uma dupla vantagem: a redução de custos e a melhoria da qualidade do serviço ao cliente. A economia de recursos está diretamente ligada à otimização da eficiência em todas as áreas operacionais, incluindo equipamentos e mão de obra. Além disso, aprimorar o atendimento ao cliente se traduz na minimização de erros e falhas no processo de separação e entrega, juntamente com uma maior agilidade em todo o ciclo de atendimento. Isso se concretiza através da combinação de melhorias no fluxo de materiais e no fluxo de informações (BANZATO, 1998) .

A implantação de um sistema como este proporciona ganhos como: aumento do nível de segurança do controle de armazenagem, dados sobre a situação do estoque, ganhos relacionados à ergonomia e organização do ambiente de trabalho (ROCHA *et al.*, 2021).

2.3 Ergonomia

A ergonomia se concentra na adaptação das condições de trabalho para alinhar-se com as habilidades dos trabalhadores. O desenvolvimento de tarefas e os equipamentos de movimentação de materiais devem levar em consideração as capacidades e restrições humanas, a fim de garantir operações seguras, reconhecendo e respeitando os limites e habilidades dos indivíduos envolvidos (CIA, 2015).

Portanto, uma das metas da ergonomia é garantir que a concepção dos sistemas de produção levem em consideração tanto a eficiência produtiva quanto o bem-estar dos funcionários. Consequentemente, os ganhos resultantes da aplicação de princípios ergonômicos em sistemas de armazenamento estão diretamente ligados à diminuição do risco de acidentes e lesões, além de melhorar as condições físicas e mentais no ambiente de trabalho. Isso, por sua vez, leva a uma drástica redução nos custos relacionados ao absenteísmo, ao seguro médico e à reabilitação (NATÁRIO, 2017).

No Brasil em especial, tem-se uma Norma Regulamentadora (NR) que diz respeito à ergonomia, a NR 17. Ela traz os requisitos mínimos para o gerenciamento dos riscos nas empresas que englobam avaliações ergonômicas com o objetivo de executar ações preventivas.

2.3.1 Norma regulamentadora 17

A NR 17 procura promover um equilíbrio entre a saúde, a segurança e o conforto para os funcionários, com o intuito de evitar que ocorram doenças causadas por esforço repetitivo. Portanto, é crucial ter um entendimento adequado dessa regulamentação, uma vez que as condições de trabalho abrangem os aspectos relacionados ao levantamento, transporte e descarga de materiais, ao mobiliário dos postos de trabalho, ao trabalho com máquinas, equipamentos e ferramentas manuais, às condições de conforto no ambiente de trabalho e à própria organização. Tudo isto estando em conformidade resulta em vantagens significativas para a saúde dos funcionários e para a imagem da empresa (MAAS *et al.*, 2020).

Ainda se tratando dos objetivos almejados, a norma visa melhorar as condições de trabalho e fazer com que todas as atividades da empresa sejam desempenhadas com maior qualidade. Para isso, o empregador deve elaborar uma análise ergonômica preliminar do ambiente de trabalho, realizada por profissional competente, para implementar ações preventivas e adaptar as condições do ambiente para que todas as atividades estejam dentro dos padrões mínimos exigidos pela norma. A aplicação da NR 17 visa uma adaptação psicofisiológica das atividades e equipamentos, diminuindo impactos causados por *stress*, cansaço ou condições que possam prejudicar a saúde ocupacional dos trabalhadores (JUNIOR, 2023).

2.3.2 Problemas relacionados à ergonomia na separação de materiais

A execução repetitiva e constante da manipulação manual de materiais em operações de logística expõe os trabalhadores ao risco de desenvolverem diversas condições, tais como Distúrbios Osteo-musculares Relacionados ao Trabalho (DORT), lesões nas costas e distensões ou entorses. As empresas estão cada vez mais conscientes dos fatores humanos envolvidos na movimentação manual de carga e nas operações logísticas internas, dedicando esforços contínuos para reduzir os riscos ergonômicos associados a trabalhos manuais repetitivos (JUNIOR, 2023).

A criação de ferramentas ergonômicas específicas para esse setor, que envolve tarefas repetitivas, é motivada pela preocupação com o desenvolvimento de distúrbios musculoesqueléticos entre os trabalhadores. A separação manual de pedidos, uma das atividades mais demoradas e desafiadoras na logística, frequentemente resulta em lesões nos trabalhadores. As pesquisas relacionadas a esses processos geralmente se concentram na eficiência de custos, com pouca consideração dada à ergonomia no design desses sistemas, o que impacta dire-

tamente nos indicadores de segurança e saúde ocupacional (GROSSE; GLOCK; NEUMANN, 2017).

A habilidade de aplicar força está diretamente ligada à postura, ou seja, é crucial utilizar os membros ou partes do corpo envolvidas na posição que proporciona a maior vantagem mecânica. Caso contrário, a eficácia dessa capacidade será significativamente reduzida. Essa dependência é tão essencial que as discrepâncias de força decorrentes de posturas inadequadas podem superar amplamente as diferenças naturais de capacidade muscular entre indivíduos. Entre os desconfortos percebidos, destacam-se as dores na região lombar, cervical e nos ombros, bem como as lesões resultantes de movimentos repetitivos das extremidades superiores do corpo humano (mãos pulso e antebraço) (NATÁRIO, 2017).

Além de abordar as posturas prejudiciais, é crucial considerar a fadiga do trabalhador, uma vez que não decorre apenas da falta de sono, mas também das características individuais dos funcionários, das nuances do trabalho, do ambiente empresarial e das condições laborais. Dentre os fatores que contribuem para a fadiga dos colaboradores, incluem-se: execução de movimentos repetitivos e manutenção de posturas estáticas por períodos prolongados; realização de tarefas monótonas; falta de informações adequadas, o que pode gerar confusão ou tédio; e exposição a carga física excessiva, especialmente em ambientes quentes. Além disso, é importante destacar que a fadiga tem repercussões significativas no sistema produtivo da empresa, podendo resultar em queda de produtividade e colocar em risco a segurança dos trabalhadores.

Estudos têm sugerido diversas alternativas para reduzir a carga física nos trabalhadores do setor logístico, especialmente em centros de distribuição. Essas abordagens atuam em três frentes principais: melhorias nos locais de trabalho, como otimização do layout das estruturas de armazenagem para facilitar o acesso ergonômico aos objetos e a criação de áreas para treinamento físico que permitam alongamentos e aquecimentos; intervenções direcionadas aos trabalhadores, como a implementação de rodízio operacional e treinamentos específicos para manejo de carga; e ajustes nos objetivos do trabalho, como o uso de veículos para reduzir a necessidade de deslocamentos a pé e monitoramento rigoroso dos pesos e disposição das cargas nas unidades de armazenamento (JUNIOR, 2023).

2.4 Otimização Combinatória

A otimização combinatória é um ramo da otimização que lida com problemas em que a solução ideal deve ser encontrada a partir de um conjunto finito de possíveis soluções discretas. Essas soluções geralmente estão em forma de combinações, permutações ou arranjos de elementos, o objetivo é encontrar uma ou mais soluções segundo uma determinada função objetivo dentro de um espaço ou região admissível definida para o problema em estudo. A palavra “combinatória” refere-se ao fato de existir apenas um número finito de soluções admissíveis e que qualquer solução tem uma propriedade combinatória que pode ser uma permutação, um arranjo, uma partição de itens, um subconjunto que corresponde a uma seleção de um conjunto

de itens (que é o caso dos problemas da mochila), árvore ou grafo que indica a relação entre os itens (CLEMENTE, 2010).

A otimização combinatória é uma área ampla e multidisciplinar que desempenha um papel crucial em resolver problemas práticos do mundo real. Diversos setores aplicam esses conceitos para melhorar processos, reduzir custos e tomar decisões mais eficientes. Exemplos de áreas onde a otimização é aplicada são: finanças, produção, escalonamento de equipes de distribuição, localizações/*layouts* de fábricas, controle de inventário, *marketing*, também estão presentes em outras áreas como empacotamento de caixas, posicionamento de satélites, alocação de trabalhadores ou máquinas e tarefas, projetos de computadores, entre outros (LUILA, 2008).

É frequente encontrarmos diversos critérios na análise de uma solução para um problema combinatório, o que confere ao problema uma natureza multi-objetiva, também conhecida como multicritério. A diversidade de soluções surge devido aos critérios serem, muitas vezes, conflitantes entre si. Ao considerarmos a compra de um automóvel, vários objetivos entram em cena. Por um lado, é desejável adquirir um veículo com excelente desempenho, medido em termos de velocidade máxima e capacidade de aceleração. Por outro lado, é essencial que o automóvel seja seguro, econômico e tenha o menor custo de aquisição possível. Essas circunstâncias tornam a resolução de problemas de otimização especialmente desafiadora, pois o número de ótimos de Pareto pode aumentar com a complexidade das instâncias envolvidas, e, em muitos casos, a determinação desses ótimos não é viável por meio das técnicas convencionais (CLEMENTE, 2010).

Para solucionar um problema de otimização combinatória pode-se caracterizar a função de maximização apresentada na Equação (1) como:

$$\begin{aligned} &\max f_1(x) = c_1 x \\ &\max f_2(x) = c_2 x \\ &. \\ &. \\ &. \\ &\max f_n(x) = c_n x \end{aligned} \tag{1}$$

sujeito à restrição:

$$x \in X = \left\{ x \in \mathbb{N}^h : x \geq 0, Ax = b, b \in \mathbb{N}^n \right\} \tag{2}$$

Em que:

c: Vetor de valores associado a função objetivo;

n: Número de funções objetivo (critérios);

h: Número de variáveis do problema de decisão;
X: Região admissível no espaço de variáveis de decisão;
x: Vetor das variáveis de decisão;
A: Matriz dos coeficientes tecnológicos ($h \times n$);
b: Vetor dos termos independentes das restrições.

A região admissível no espaço das funções objetivo (o conjunto de todas as imagens dos pontos em X), isto é, o espaço dos critérios de otimização, pode ser definida da seguinte forma:

$$F = \{f \in \mathbb{N}^n : f = (f_1(x), f_2(x), \cdots, f_n(x)), x \in X\} \quad (3)$$

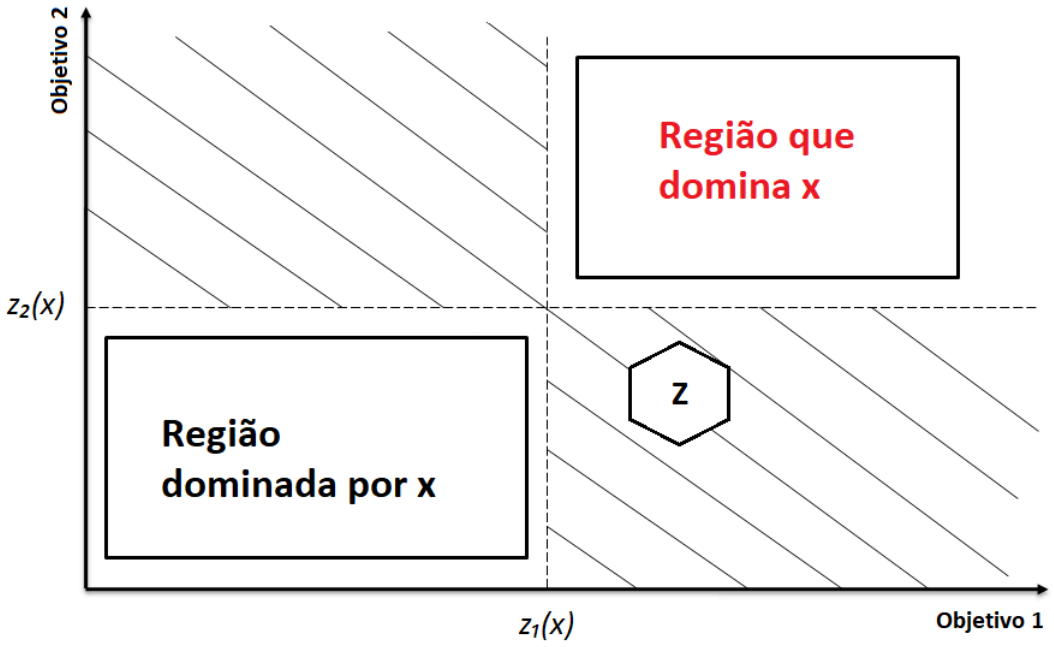
Baseado no que diz a Equação (3), a função objetivo representa o que se está tentando otimizar. Cada função componente $f_n(x)$ no seu conjunto F representa uma dessas funções objetivo. No contexto da resolução de problemas de otimização multicritério, o desafio consiste em identificar o conjunto abrangente de soluções viáveis que "otimizem" o vetor de funções objetivo do problema. Dentro da região admissível, onde várias soluções podem coexistir, é crucial compreender o tipo de soluções desejadas. Em outras palavras, é fundamental compreender a definição do conjunto de soluções admissíveis que busquem o vetor ótimo das funções objetivo, introduzindo assim o conceito fundamental da dominância na otimização multicritério (LUILA, 2008).

Enquanto na resolução de problemas com um único objetivo busca-se a solução ótima, ou seja, a solução admissível que maximiza ou minimiza a função objetivo única, essa abordagem não é aplicável em problemas multicritério. Em cenários multifacetados, em que conflitos entre objetivos são inevitáveis, uma solução admissível que otimiza um dos objetivos geralmente não otimiza os demais. Portanto, torna-se essencial compreender e aplicar o conceito de dominância ao lidar com problemas de otimização multicritério (CLEMENTE, 2010).

A dominância pode ser frequentemente utilizada para comparar e avaliar soluções em relação a múltiplos critérios. A dominância é uma relação de ordem parcial que ajuda a identificar quais soluções são melhores que outras em termos de todos os critérios considerados. Pode-se supor um problema onde existem diversas soluções com vários critérios de otimização e suas respectivas funções objetivo, pode-se comparar uma solução Z_1 a uma solução Z_2. Z_1 irá dominar Z_2 quando Z_1 for maior ou igual a Z_2, sendo o primeiro uma função de x e o segundo uma função de y. Pode-se observar na Figura 4 o comparativo das regiões e funções em Z, sendo que a solução em x domina uma região e é dominada em outra e o restante considerado solução do sistema (LUILA, 2008).

Os tipos de soluções existentes para a otimização, em geral são: solução eficiente e solução não-dominada. As soluções não-dominadas (ou Fronteira de Pareto), caracterizam-se pelo fato de que não se pode melhorar todos os critérios ao mesmo tempo. A solução não pode ser melhorada com relação a um objetivo sem piorar sua resposta com relação a algum

Figura 4 – Dominância e regiões



Fonte: Adaptado de (LUILA, 2008).

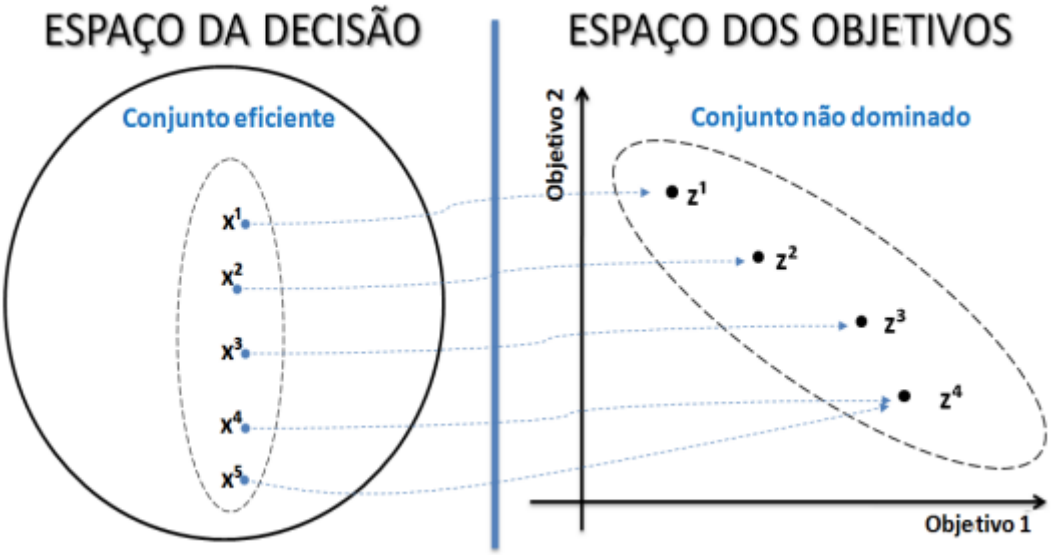
outro. Isto traz a ideia de que nem todas as soluções são igualmente boas, pois algumas podem ser superiores em alguns pontos e inferiores em outros. Uma solução eficiente x pertence ao conjunto X, é considerada eficiente quando não é possível encontrar outra solução admissível y em X, que seja pelo menos tão boa quanto x em todos os critérios e seja estritamente superior a x em pelo menos um critério. Em outras palavras, x é uma solução eficiente se não houver outra solução admissível y em X que supere x em todos os critérios (CLEMENTE, 2010).

A ideia de eficiência normalmente é aplicada a pontos no espaço de decisão, ao passo que o conceito de não dominância é empregado para descrever a correspondente representação no espaço das funções objetivas. Em outras palavras, uma solução que não é dominada representa a eficiência de uma solução (LUILA, 2008), isto pode ser visto na Figura 5.

2.4.1 Problema da Mochila

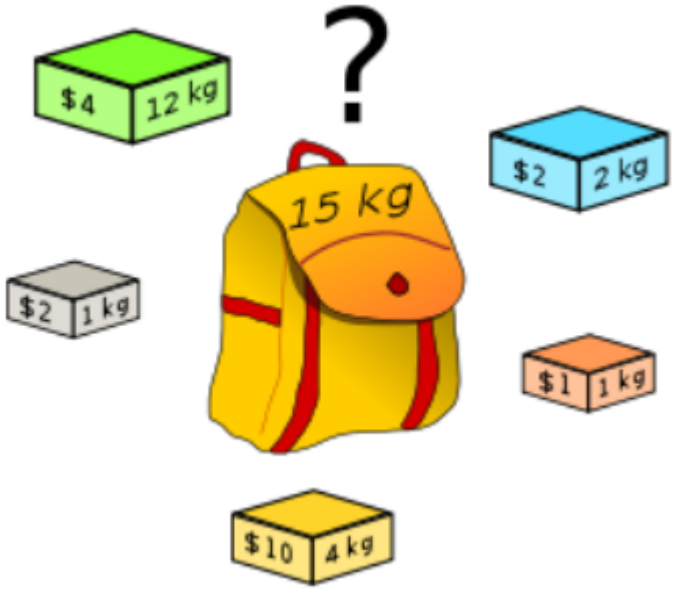
O problema da mochila recebe este nome devido a uma analogia que se faz com a necessidade de se preencher uma mochila com objetos de diferentes pesos e valores (custo). O objetivo é maximizar o valor total, sem ultrapassar o peso máximo permitido. Os desafios relacionados às mochilas são comumente representados por modelos de programação inteira, nos quais parte ou a totalidade das variáveis envolvidas no problema são restritas ao conjunto dos números inteiros. Uma ilustração deste problema pode ser observada na Figura 6 (VERÇOSA, 2019).

Figura 5 – Eficiência e não dominadas



Fonte: Adaptado de (CLEMENTE, 2010).

Figura 6 – Exemplo de problema da mochila



Fonte: (AMARANTE, 2013).

No âmbito da Programação Linear Inteira Mista, o desafio da mochila é um clássico amplamente examinado. Sua importância histórica decorre do fato de que: (i) é considerado o mais simples entre os problemas de MIP; (ii) emerge como um subproblema em diversos contextos mais complexos; e (iii) abrange uma ampla gama de situações reais e práticas, deste modo considera-se um conjunto contendo n itens, onde cada item j em $(j = 1, \cdots, n)$ possui um peso w_j e um lucro v_j associado. O lucro é uma medida que indica a importância do objeto, sendo fundamental para a priorização dos itens a serem incluídos na mochila. A mochila tem uma capacidade C, que representa o peso máximo que ela pode suportar. A variável x_j

é empregada para indicar se um item específico j foi inserido na mochila. Quando o item j é escolhido para ser colocado na mochila, a variável x_j é atribuída o valor 1; caso contrário, recebe o valor 0. O objetivo da resolução desse problema é maximizar o lucro obtido pelos itens que são inseridos na mochila (MARTELLO; TOTH, 1990).

$$\max z = \sum_{j=1}^{n} v_j x_j, \tag{4}$$

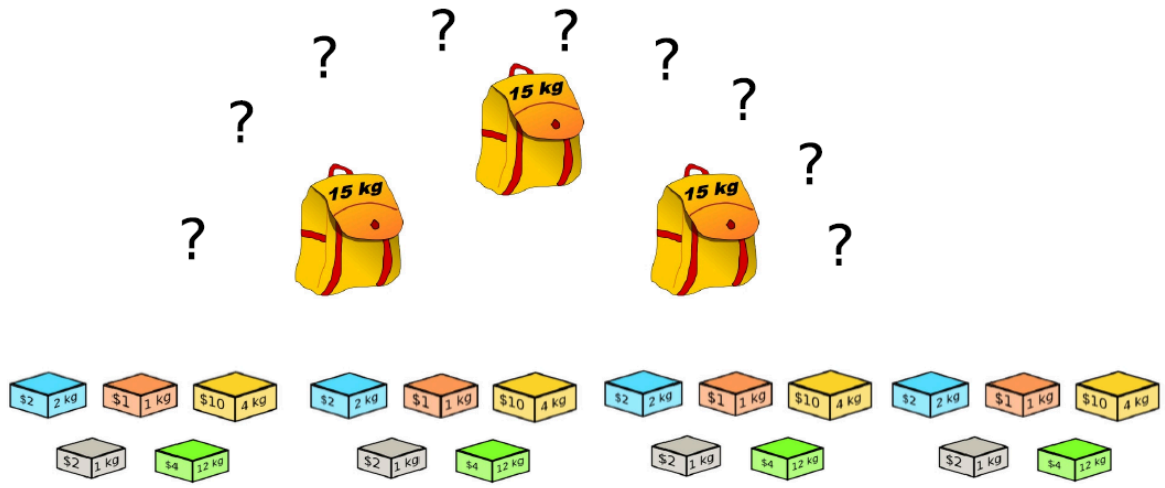
sujeito à restrição:

$$\sum_{j=1}^{n} w_j x_j \leq C, \tag{5}$$

$$x_j = 0 \text{ ou } 1, j \in N = \{1, \cdots, n\} \tag{6}$$

A Equação (4) tem a função de maximizar o lucro decorrente dos itens inseridos na mochila. Já o termo descrito na Equação (5) apresenta uma restrição do sistema onde a soma dos pesos de todos os itens a serem inseridos na mochila não pode ultrapassar a capacidade de carga C da mesma. Adicionalmente a esta primeira restrição declarada é indicada também outra na Equação (6) que diz: especifica-se que a variável x_j pode somente receber (0 ou 1), o que informa ao sistema se o produto já foi ou não inserido. Outra aplicação característica é o problema da mochila com múltiplas mochilas. Este problema está representado na Figura 7. A distinção com o problema anterior consiste na quantidade de mochilas empregadas: ao passo que no cenário mais básico apenas uma mochila é utilizada, no contexto de múltiplas mochilas são empregadas diversas delas. Essa questão foi concebida para ser aplicada em situações que envolvem tolerância a falhas, esquemas de criptografia pública, alocação e escalonamento, entre outras aplicações (AMARANTE, 2013).

Figura 7 – Exemplo de problema da mochila múltipla



Fonte: (AMARANTE, 2013).

Matematicamente pode-se definir este problema como:

$$\max z = \sum_{i=1}^{m} \sum_{j=1}^{n} v_j x_{ij}, \tag{7}$$

sujeito às restrições:

$$\sum_{j=1}^{n} w_j x_{ij} \leq c_i, \quad i \in M = \{1, \cdots, m\}, \tag{8}$$

$$\sum_{i=1}^{m} x_{ij} \leq 1, \quad j \in N = \{1, \cdots, n\}, \tag{9}$$

$$w_j \leq \max_{i \in M} \{c_i\}, \quad \text{para } j \in N, \tag{10}$$

$$c_i \geq \min_{j \in N} \{w_j\}, \quad \text{para } i \in M, \tag{11}$$

$$v_j \; e \; c_i \rightarrow \text{ inteiros positivos}, \tag{12}$$

$$x_{ij} = 0 \text{ ou } 1, \quad i \in M, j \in N \tag{13}$$

Em que:
n: Quantidade total de itens;
m: Quantidade total de mochilas;
$w_j \; e \; v_j$: Representam, respectivamente, o peso e o valor do item j;
c_i: Representa a capacidade da mochila i;
x_{ij}: Informa se o item j está presente ($x_{ij} = 1$) ou não ($x_{ij} = 0$) na mochila i;

O propósito desta solução, apresentado na Equação (7) é otimizar o valor total alcançado ao alocar os elementos nas mochilas. As restrições impostas incluem a Equação (8) e a Equação (9), que estipulam, respectivamente, que a soma dos pesos dos itens em uma mochila não pode ultrapassar a capacidade total da mochila, e que um item só pode ser inserido em uma única mochila. É essencial também estabelecer requisitos mínimos para a inserção de um item nas mochilas, o que implica na aplicação de regras que limitam o peso máximo de um item a ser menor ou igual à maior capacidade entre todas as mochilas, conforme a Equação (10), e que a capacidade de uma mochila deve ser maior ou igual ao menor peso entre os itens da Equação (11). Por fim, a restrição apresentada na Equação (12) restringe o problema a lidar apenas com valores inteiros e positivos (AMARANTE, 2013).

Dada a formulação do problema conforme previamente descrito, torna-se imperativo empregar uma abordagem para a sua resolução. Nesse contexto, optou-se por utilizar o algoritmo fundamentado pela programação linear inteira mista, com a utilização de solucionadores.

2.4.2 Algoritmo e Solucionador

Os solucionadores SCIP e GLOP são utilizados para a resolução de problemas de programação linear e não linear com restrições mistas. Eles possuem código aberto e são projetados para resolver problemas complexos de otimização que podem envolver também variáveis de decisão discretas.

O *Solving Constraint Integer Programs* SCIP utiliza um algoritmo *branch-and-bound* para explorar o espaço de soluções. Este método envolve decompor o problema em subproblemas para encontrar a solução ótima. Ele suporta também a computação paralela, permitindo tirar vantagem de vários processadores ou núcleos para resolver problemas de otimização de forma mais eficiente. O SCIP é projetado para ser extensível, permitindo também que sejam aplicadas heurísticas personalizadas, regras de ramificação e outras características para adaptar o solucionador aos determinados problemas.

O GLOP é um solucionador de otimização linear desenvolvido pelo *Google*. O algoritmo utilizado internamente a este *solver* é o *Simplex* Duplo. Este algoritmo caminha ao longo dos vértices dentro da região possível para a resposta, desta maneira melhora de forma iterativa o valor da função objetivo até a solução ideal. Sua utilização se dá principalmente em algoritmos MIP (OR-TOOLS, 2023).

O CP-SAT combina a resolução do *solver* SAT e do *solver* de programação de restrições puramente integral, usando a geração de cláusulas preguiçosas. O modelo *simplex* contido no CP-SAT traz vantagens óbvias de uma relaxação linear na parte do modelo completo. Ele apresenta resultados inovadores em *benchmarks* de escalonamento e resultados competitivos com os outros solucionadores (em problemas puramente integrais (PERRON; DIDIER; GAY, 2023).

3 ESTADO DA ARTE

A solução de diversos problemas atuais utilizando a adequação ao problema da mochila e de múltiplas mochilas tem sido estudadas em diversas pesquisas (LUILA, 2008). Alguns estudos adaptam o problema da mochila à computação nas nuvens, utilizando-o para definir o melhor planejamento da arquitetura de um *data center* a ser utilizado e para chegar a este objetivo utiliza-se da modelagem do problema de alocação de máquinas virtuais e de um algoritmo baseado em colônia de formigas para solucionar os problemas encontrados (AMARANTE, 2013).

O problema da mochila clássico é caracterizado por possuir somente duas variáveis para análise como peso e valor de utilização ou custo. Inúmeros desafios do mundo real podem ser abordados utilizando o conceito do problema da mochila ou suas variantes, tais como: questões de otimização no corte de materiais, controle de orçamento, embalagem eficiente, distribuição de cargas em equipamentos e seleção criteriosa de projetos de investimento (CLEMENTE, 2010).

Em uma das aplicações analisadas, (BRETAS, 2016) apresentou um modelo matemático baseado no problema da mochila para decidir na compra de aparelhos destinados à atividades físicas. O projeto justifica-se como uma otimização dos espaços, com o maior número de equipamentos, em concordância com os espaços destinados para a circulação de pessoas. Para este projeto foi considerado o problema da mochila inteiro clássico como fundamentação básica para elaboração do modelo matemático. Através de uma função objetivo considerando suas restrições, ele soluciona o problema e encontra um resultado. A resposta obtida se mostra satisfatória, indicando os itens a adquirir. Todos os resultados observaram as restrições indicadas no início, como limitantes para a execução do projeto.

No caso apresentado por (MARQUES; ARENALES, 2002) foi proposto um modelo de otimização não linear inteiro para o problema e algumas heurísticas para sua solução. Dentro delas, duas são as mais significativas para o estudo sendo elas: Decomposição e Melhor Compartimento. A primeira é seccionada em duas etapas, inicialmente resolvendo o problema com relação à capacidade, o que resulta nos melhores compartimentos relacionados aos agrupamentos e na segunda etapa, um problema da mochila clássico é resolvido, considerando os compartimentos obtidos na primeira etapa como super itens. A segunda heurística, como na anterior, o melhor compartimento para os itens é determinado, em seguida é selecionado o compartimento de maior valor por unidade, o qual é alocado na mochila. O processo é repetido até que tudo seja incluído na mochila.

O trabalho de (AMARANTE, 2013) buscou solucionar o problema de múltiplas mochilas que está relacionado à alocação usando como base o algoritmo que corresponde à metaheurística de otimização por colônia de formigas. Ele identifica o conjunto de máquinas virtuais e de servidores. Uma máquina virtual requisita uma certa quantidade de recursos. Já o servidor deve monitorar a quantidade de recursos que ainda possui e o que já foi utilizado. Para o fun-

cionamento do algoritmo, deve ser passado como parâmetro o conjunto de máquinas virtuais a serem alocadas e o vetor C de capacidade dos servidores. Como resultado final do trabalho, (AMARANTE, 2013) comparou os resultados do algoritmo aplicado ao problema da mochila com outras heurísticas já consideradas para o problema de alocação das máquinas virtuais, com relação ao consumo de energia e tempo de processamento de um conjunto de tarefas.

Tem-se ainda uma abordagem à solução do problema da mochila quadrático utilizando uma aproximação com um algoritmo de otimização linear mista. Esta abordagem foi proposta por (CUNHA; SIMONETTI; LUCENA, 2014). Neste trabalho é abordado um conjunto de n objetos com pesos W, uma mochila com capacidade C, onde deve-se colocar na mochila um objeto i que traz um benefício P e deseja-se escolher um subconjunto de objetos N' carregados na mesma mochila. O autor interpreta o problema da mochila como os termos da Teoria dos Grafos. O algoritmo utilizado para resolver o problema é conhecido como *Branch and Cut*. Este problema caracteriza-se por nós de uma árvore de enumeração e para eles são gerados planos de cortes para diferentes problemas encontrados. Dependendo do número de cortes, a demanda relativa ao uso de memória *Random Access Memory* (RAM) pode se tornar significativa. O que permitiu a utilização do algoritmo foi um resolvedor de programação linear inteira mista: o *Xpress Optimizer*. A variável escolhida para ramificação na árvore, foi escolhida pelo resolvedor automaticamente. O algoritmo demonstra eficácia ao resolver o sistema, no entanto, ao incorporar as desigualdades, revela-se computacionalmente custoso, apresentando ainda um número elevado de nós nas árvores de enumeração.

Outro trabalho que auxiliou na pesquisa e motivação desta dissertação foi o de (VERÇOSA, 2019). Ele propôs em seu projeto o modelo e algoritmos para seleção de sensores como serviço. A *Internet of Things* (IoT) ou internet das coisas impulsiona o desenvolvimento de novas tecnologias e essas tecnologias cada vez mais conectadas. Neste cenário a aquisição de dados se torna uma grande aliada na tomada de decisões que deve ser realizada dependendo de cada caso. Para esta finalidade que existem os sensores, que podem ser se vários tipos (como temperatura, pressão, umidade, velocidade do vento). O trabalho apresenta um modelo matemático para a seleção de sensores capaz de maximizar o atendimento aos requisitos de entrada do usuário, como precisão e robustez.

Considerando o modelo baseado no problema da mochila, é aplicado o método *Branch and Bound* que é muito utilizado para encontrar soluções ótimas para problemas de programação inteira e discreta apicados à análise combinatória. O problema pode também ser comparado a uma árvore de pesquisa implícita ou explicita. Cada nó da árvore pode ser uma solução parcial ou completa para o sistema. Em especial neste trabalho, (VERÇOSA, 2019) utilizou um *solver* chamado *Gurobi*. Ele inicia pelo MIP (*Mixed Integer Programming*) original, onde no caso do algoritmo não conseguir solucionar a demanda, as restrições devem ser removidas, isto é chamado de relaxamento da programação linear do MIP. O modelo matemático descrito no trabalho então é resolvido e são testadas todas as restrições para solucionar completamente o problema.

Novamente, como citado em trabalhos relacionados, o autor utiliza uma função objetivo como ponto de partida.

A função objetivo busca maximizar o atendimento dos requisitos do usuário com os sensores selecionados. A seleção final dos sensores é realizada pela definição das variáveis de decisão. A primeira restrição estabelece um principio orçamentário ao problema. Outra restrição garante que o sistema contenha o número exato de tipos de sensores estabelecidos pelo usuário. Para verificar se o modelo atende aos requisitos, um índice de atendimento foi avaliado assumindo que cada sensor é um ponto em um espaço multidimensional de propriedades (localização, robustez, precisão e frequência de coleta).

Como conclusão, o autor convergiu o trabalho para uma implementação de soluções, usando o *solver*, que em seu estado atual, permitiu toda a tomada de decisão quanto a seleção dos sensores de maneira eficaz. O algoritmo foi avaliado através de simulações *Monte Carlo*, com a análise baseada em três métricas: tempo de otimização, valor da função obtido na solução ótima e orçamento utilizado na solução ótima (VERÇOSA, 2019).

O trabalho de (LUILA, 2008) aborda o problema da mochila multicritério considerando aspectos algorítmicos e implementação informática. O objetivo de seu estudo foi de explorar de maneira eficaz o algoritmo de etiquetagem em uma abordagem multicritério e visa resolver instâncias do problema da mochila, tornando-as cada vez maiores em um intervalo de tempo reduzido, devido à crescente demanda por algoritmos eficientes capazes de calcular soluções não dominadas de forma rápida. O autor aborda diversos problemas da mochila diferentes indicando suas equações matemáticas e as diversas utilizações.

Ao longo do texto são apresentadas metodologias para resolver os problemas de multicritério que são: Método de agregação a *priori* de preferências, métodos geradores das soluções eficientes, onde não há articulação de preferências e método de “articulação progressiva de preferências” (interativos). No primeiro método, ao identificar as preferências, é viável converter o problema inicial em um de único critério, usando uma função de utilidade. Apesar da modelagem inicial ser multicritério, o foco está em otimizar a função de utilidade, cuja solução ótima será a final. No segundo método são calculadas todas as soluções eficientes do problema e colocadas à disposição para serem avaliadas. Já no terceiro método são expressas as preferências através de um processo, de maneira que a pesquisa seja conduzida para a região onde se encontram as melhores soluções, este último é utilizado na implementação do algoritmo de etiquetagem (LUILA, 2008).

Por fim, (LUILA, 2008) apresenta os resultados obtidos pela aplicação do algoritmo e o seu comportamento computacional. As diferentes instâncias consideradas possuíam objetivos e restrições diferentes, com capacidades da mochila também diferentes. A abordagem do algoritmo se revelou eficiente para resolver problemas multicritério de pequenas dimensões. Porém, quando exposto para problemas complexos, a memória utilizada condicionou o desempenho do algoritmo, apresentando uma clara limitação. Em projetos futuros, conforme diz o autor, indicará uma gestão mais eficiente da memória computacional.

Seguindo ainda pela mesma linha de pesquisa dos problemas da mochila multicritério, outro autor que contribuiu para fomentar este trabalho é (CLEMENTE, 2010). A pesquisa que o autor realizou tem como objetivo implementar uma metodologia hibrida, ou seja, o método da pesquisa por dispersão (interativa) combinado com um algoritmo da colônia de formigas para solucionar o problema da mochila multicritério com uma ou várias restrições.

Na aplicação da colônia de formigas ao problema da mochila, em cada iteração (ciclo), uma formiga artificial decide qual caminho a seguir, escolhendo os vértices mais atrativos, dentre os não selecionados. Cada uma das formigas possui uma memória que armazena o percurso e não permite que seja escolhido mais de uma vez, ou seja, cada formiga traz para a mochila a cada iteração um item, respeitando a restrição imposta. A seleção de um vértice, isto é, um item $j \in \{1, \cdots, n\}$, obedece a uma regra que resulta da ponderação da quantidade de feromonas e da informação heurística associada a este vértice. Associado a este algoritmo é apresentada a meta-heurística da pesquisa por dispersão, que trata-se de um método evolucionário e obtém novas soluções através da combinação de conjunto de soluções (CLEMENTE, 2010).

Ao final do trabalho (CLEMENTE, 2010) definiu os aspectos fundamentais dos modelos informáticos apresentados, a estrutura de dados de entrada e saída, os parâmetros algorítmicos utilizados em cada variante. Como métrica, foram utilizados indicadores para medir o desempenho de cada variante para realizar uma comparação dos modelos implementados de acordo com as medidas de avaliação utilizadas. Ainda na finalização do capítulo são comparados os modelos implementados e sua aplicação ao problema da mochila, bem como se as variantes do algoritmo estão de acordo com outros métodos evolucionários encontrados na literatura.

Outro trabalho relacionado ao problema da mochila é apresentado por (NETO; PEDROTTI; HOSHINO, 2023) que busca uma resolução para os problemas de roteamento de veículos. Estes desafios consistem em determinar rotas para cada veículo de uma frota homogênea e limitada, de uma maneira que todas as demandas de todos os clientes sejam atendidas por meio da entrega de produtos a partir de um depósito central com o menor custo possível. Cada rota serve a diferentes grupos de clientes, sem sobreposição, e a soma das demandas dos clientes atendidos por cada rota deve respeitar a capacidade máxima de carga de cada veículo.

O objetivo é avaliar o uso de meta-heurísticas, que combinam programação matemática e meta-heurísticas, para resolver os problemas. A única meta-heurística existente para esses problemas é a heurística de geração de colunas para o *Capacitated Profitable Tour Problem*. O estudo analisa as técnicas *Relax&Fix* e *Large Neighbourhood Search*, utilizando um modelo estendido de programação linear inteira. Um algoritmo *Branch&Price* exato foi implementado, utilizando a relaxação “ng-route” como subproblema de precificação. A resolução da relaxação linear pelo método de geração de colunas permite construir o problema da mochila com restrição de conflitos, onde as rotas dos veículos são interpretadas como itens que não podem visitar o mesmo cliente. A capacidade da mochila representa o total de veículos da frota, e o valor de cada item é a soma dos prêmios dos clientes atendidos menos o custo da rota.

Segundo (NETO; PEDROTTI; HOSHINO, 2023) os experimentos computacionais foram realizados em instâncias da literatura. As meta-heurísticas foram implementadas na linguagem C, compiladas com gcc 4.9.2 e usam a biblioteca de otimização SCIP (v3.2.1). Os experimentos mostram que a combinação de *Relax&Fix* e *Large Neighbourhood Search* no algoritmo exato produziu os resultados mais promissores entre as meta-heurísticas propostas. No entanto, é possível melhorar os algoritmos, especialmente ajustando alguns parâmetros do "ng-route" na geração de colunas, que afetam a qualidade das rotas usadas pelas heurísticas. Conjuntos ng maiores tornam as rotas mais elementares e melhoram a qualidade dos limitantes, embora aumentem o tempo de processamento. Além disso, rotinas para converter rotas não elementares em elementares também podem melhorar o desempenho das meta-heurísticas.

Ligados à solução com heurísticas (JANUÁRIO, 2023) utiliza a busca em vizinhanças variáveis como solução para o problema da mochila multidimensional. O problema encontra-se na categoria de programação linear inteira binária, onde as restrições e a função objetivo são lineares e os valores para cada resolução do problema devem ser binários inteiros 0 ou 1. Algumas técnicas para a resolução deste desafio incluem programação dinâmica e *branch & bound*.

A meta-heurística estudada e implementada ao longo do desenvolvimento deste trabalho foi executada com as mesmas instâncias de testes usadas por diversos trabalhos de referência para possíveis efeitos comparativos. (JANUÁRIO, 2023) aborda a técnica da busca de vizinhança variável, que consiste em realizar trocas sistemáticas entre vizinhanças para explorar o espaço de busca do problema. Enquanto o algoritmo é executado, é escolhido supostamente o melhor local calculado pelas buscas locais, sendo considerado o ótimo global. Nesta busca, parte-se de uma solução inicial gerada s e escolhe-se aleatoriamente um vizinho s', na sequência, utiliza-se uma estratégia de busca local dando como entrada a solução s' e armazenando o resultado obtido em s".Se o valor da função objetivo do novo resultado encontrado pela busca for maior que o armazenado em s, reinicia-se a busca na primeira vizinhança, pois agora há uma solução superior. Caso contrário, apenas ocorre a troca para a próxima vizinhança. Ao adaptar para o trabalho, o autor expandiu a frequência com que o algoritmo utiliza as estruturas da vizinhança, visando explorar mais o espaço de busca, fazendo com que tenha mais iterações.

Segundo (JANUÁRIO, 2023) os resultados obtidos foram de boa qualidade. Com o algoritmo proposto observou-se um aumento no número de vizinhanças e também no tempo médio de execução de algumas instâncias. Portanto, vê-se necessária a busca por novas técnicas para reduzir este tempo de execução.

Outro trabalho que aborda uma resolução do problema da mochila multidimensional binária é (YAMASHITA; GARCIA, 2023) que analisa a aplicação do algoritmo colônia de abelhas artificiais para a resolução. Nesse algoritmo iterativo, a colônia é simplificada em classes de abelhas que podem ser consideradas como: empregadas, observadoras e escoteiras. A melhor solução para o problema é a busca por alimento para a colônia e este será sempre o objetivo final considerado para todas as abelhas.

Na natureza a atividade de forrageamento das abelhas é essencial à sua sobrevivência. As abelhas exploram o ambiente em busca de alimento (pólen, néctar e água). Na divisão das classes, as empregadas, buscam solucionar o problema atuando em uma determinada fonte já conhecida de alimento. As observadoras devem escolher a fonte de alimento com base nas aptidões das soluções apontadas pelas abelhas empregadas. Já as escoteiras devem explorar novas soluções artificialmente caso haja falha nas soluções apontadas anteriormente. A inicialização da população é feita de forma aleatória, porém deve ser observado que toda a população não deve ultrapassar as restrições de recursos da mochila.

O algoritmo opera por meio de ciclos globais, durante os quais as abelhas observadoras e escoteiras exploram suas fontes. O limite global refere-se ao número máximo de ciclos nos quais as soluções iniciais podem ser aprimoradas. Isso significa que, após a inicialização das abelhas, elas têm até o limite global de tentativas para encontrar a solução ótima do problema dado. Se não conseguirem dentro desse prazo, o melhor resultado obtido é registrado e o algoritmo segue para a próxima execução.

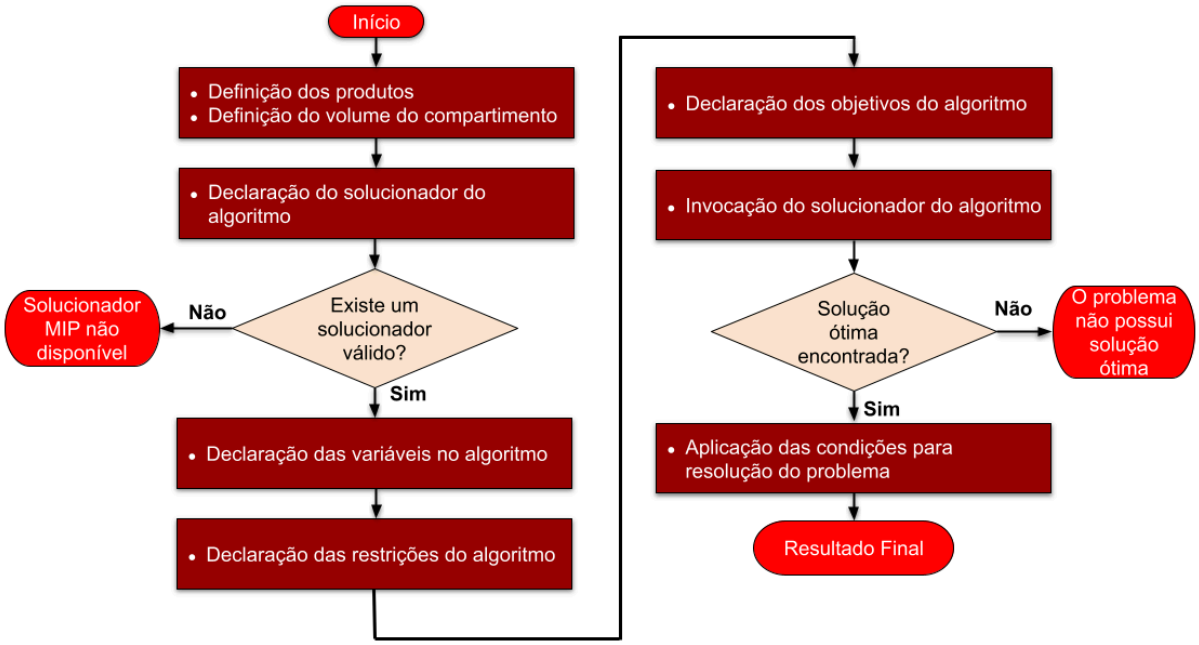
Como maneira de avaliar o trabalho proposto (YAMASHITA; GARCIA, 2023) utilizou três diferentes conjuntos de testes encontrados na literatura, entre eles: o conjunto *SAC-94*, a biblioteca *ORLIB* e o conjunto *GK*. O algoritmo demonstrou uma grande eficácia na resolução de problemas da biblioteca *SAC*. Neste estudo pode-se concluir que quanto maior o número de iterações, mais robusto é o algoritmo aumentando o número de soluções ótimas, ou próximas das ótimas encontradas.

Todos os trabalhos apresentados contribuíram para uma visualização do cenário de pesquisa, no que diz respeito ao estudo do problema da mochila e suas variações. Observa-se que diversas aplicações do mundo atual podem ser caracterizadas como um problema da mochila. Observa-se também as soluções para sua resolução, seja utilizando técnicas de otimização linear inteira ou mista ou utilizando meta-heurísticas adaptadas ao objetivos específicos.

4 MATERIAIS E MÉTODOS

O presente capítulo aborda quais os materiais e métodos adotados para a realização do trabalho. A metodologia proposta foi aplicada em uma empresa do ramo de estruturas de armazenagem e automação de transportadores motorizados, localizada na região de Ponta Grossa, Paraná. Para a elaboração dos dados e restrições do projeto é necessário o uso do *software Microsoft Excel* para confecção de tabelas e cálculos. Para a elaboração e aplicação do algoritmo de otimização na resolução do projeto é utilizado o *software Microsoft Visual Studio Code* (VSC). Este *software* permite a escrita dos programas utilizando uma linguagem de programação, onde fornece suporte interno para *JavaScript*, *TypeScript* e *Node.js* e possui uma gama de extensões e aplicações para outras como: *C++, C#, Java, Python, PHP, Go, .NET*. Para a implementação deste projeto utilizou-se a linguagem *Python*. Além do desenvolvimento dos programas, o VSC permite a realização do *debug* deste código a fim de visualizar se as saídas estão correspondentes às expectativas do programador.

Figura 8 – Fluxograma do Algoritmo



Fonte: Autoria própria (2024).

As etapas do desenvolvimento deste trabalho estão de acordo ao que é indicado no fluxograma representado na Figura 8, para ambos os solucionadores utilizados. Para realizar as aplicações da otimização sobre o problema encontrado é necessário delimitar alguns pontos, que indicam os produtos e as limitações. Iniciando este desenvolvimento, deve-se estabelecer onde serão alocados os produtos considerados para o projeto.

A ideia do trabalho é a aplicação em uma estrutura de *Flow-rack* que é dividida em compartimentos. Inicialmente deve-se conhecer o que será armazenado. Com a tabela e os produtos definidos, deve-se delimitar a capacidade dos compartimentos, também chamados

bins. Com as dimensões do *bin* definidas, é possível calcular o seu volume. Este é um dado fundamental para alimentar o algoritmo utilizado onde a alocação destes produtos não poderá exceder o volume total do compartimento.

Cada compartimento possui uma capacidade máxima que deve ser preenchida de acordo com critérios de frequência de saída dos produtos e posição ergonômica que facilite a retirada e reduza os impactos sobre a qualidade de vida dos operadores, reduzindo a necessidade de afastamento dos mesmos por motivo de doença relacionada ao trabalho. Os compartimentos são numerados para que se possa avaliar com o algoritmo se a distribuição dos produtos está atendendo aos critérios de projeto indicados inicialmente.

$$V = A * L * C \tag{14}$$

onde V é o volume, A é a altura, L a largura e C o comprimento.

Aplica-se a Equação (14) também para todos os produtos definindo o volume ocupado por cada um. Isto se faz necessário para alocar corretamente nos compartimentos.

Como o problema da mochila utilizado neste projeto é o problema da mochila multicritérios, deve-se considerar mais uma variável para definir as ocupações corretas nos *bins*.

A frequência de seleção dos produtos é definida pelo sistema através de uma média histórica de separação. Para o projeto, as porcentagens foram arbitradas de maneira aleatória, porém de modo uniforme. Dentre os produtos escolhidos, a frequência é dividida até alcançar os 100%. De acordo com as distribuições apresentadas, foi aplicado o algoritmo para a solução de um problema de MIP.

O pacote de *software* do algoritmo utilizado para a resolução do problema é conhecido como *OR-Tools*, que possui código aberto para otimização, ajustado para enfrentar os problemas mais difíceis em roteamento de veículos, fluxos, programação linear e programação de restrição (OR-TOOLS, 2023).

Para a aplicação deste modelo necessita-se que as restrições sejam lineares. Nesta fase do trabalho são implementados os algoritmos SCIP, GLOP e CP-SAT. As etapas básicas de aplicação de todos os *solvers* são as descritas no fluxograma da Figura 8 e podem ser detalhadas como declaração do *Wrapper*, declaração das variáveis de entrada e criação das variáveis do sistema.

A declaração do *Wrapper* consiste em aplicar o *solver* desejado, para isto é necessário declarar a biblioteca do *OR-Tools*. Isso tornam disponíveis todas as funções e *solvers*, que permite aplicar técnicas para alocar os produtos de maneira automática, porém a divisão de produtos por compartimentos ainda deixa a desejar no quesito espalhamento de produto nas extremidades do *Flow-rack* e a falta de observação de melhores condições ergonômicas para o desempenho do operador.

A declaração das variáveis de entrada vem logo pós a definição da biblioteca de otimização, devem-se declarar as variáveis do sistema que serão manipuladas pelo algoritmo para se

chegar no objetivo final. Como este trabalho é voltado para a resolução do problema de alocação de produtos, o enfoque maior se dá para o volume que os produtos ocupam no compartimento e seu custo. Especificamente neste projeto utiliza-se no lugar do custo, a variável "frequência".

Depois de definidas as variáveis de entrada, é necessária a criação das variáveis do sistema para informá-lo de como ele deve considerar essas informações, correlacionando-as entre si. Isto se dá por um algoritmo que combina os vetores em posições com linhas e colunas. No Algoritmo 1 pode ser visualizado um exemplo de código fonte.

Algoritmo 1 – Exemplo do código para a criação de variáveis.

```
1:
2: x[i,b] = 1 if item i is packed in bin b
3: x = {}
4: para i in data["all_items"] : faça
5:     para b in data["all_bins"] : faça
6:         x{i,b} = solver.BoolVar(f"x_{i}_{b}")
7:     finaliza para
8: finaliza para
```

Fonte: Autoria própria (2024).

Considerando o Algoritmo 1, que mostra um algoritmo genérico somente a título de exemplo, cada $x[(i,j)]$ é uma variável 0-1, em que i é um item e j é um agrupamento. Na solução, $x[(i,j)]$ será 1 se o item i for colocado no compartimento j e 0 caso contrário, relacionando sempre um item a um compartimento.

A alocação dos produtos em um problema multiobjetivo deve seguir mais de um critério e isto pode ser um problema muitas vezes complexo. As restrições do algoritmo podem ser consideradas como: restrição de volume e restrição de quantidade. As restrições são definidas para limitar as escolhas dos produtos de acordo com certas características impostas pelo problema.

Na restrição de volume, os volumes dos objetos são somados e seu resultado não pode exceder um valor limite para a mochila. Já na restrição de quantidade, indica que um objeto pode ser alocado somente em um compartimento. O algoritmo então deve buscar alcançar a função objetivo sem ferir nenhuma das restrições.

A definição dos objetivos é baseada na frequência, de acordo com a frequência com que o objeto é selecionado pelo operador (pode-se assemelhar esse parâmetro ao custo), o valor é maximizado e dividido em mochilas diferentes de maneira a diversificar ou concentrar determinados produtos em diversos compartimentos.

Por fim, assim que realizada a criação do solucionador, o mesmo é acionado de acordo com o modelo definido previamente e o resultado da solução é tratado e mostrado no terminal de saída dos dados. A partir desse ponto, os dados podem ser analisados e manipulados para entender se a solução é suficiente ou se outros solucionadores devem ser explorados indicando uma distribuição com maior assertividade com relação ao objetivo.

5 RESULTADOS

Este capítulo apresenta os resultados do trabalho envolvendo coleta de dados, tratamento dos dados, a aplicação do algoritmo utilizando os valores dos produtos analisados e por fim, a resposta do sistema e sua saída.

O *Flow-rack* que ficou definido para este projeto irá possuir 12 *bins*, divididos em 3 linhas, sendo 4 compartimentos por linha, conforme Figura 9.

Figura 9 – Exemplo de *Flow-rack* do caso

Fonte: Autoria própria (2024).

Para a definição dos itens, consultou-se os produtos de mercado voltados à aplicação na indústria farmacêutica, validando as dimensões de cada um deles e incluindo os mesmos numa tabela. Para atender todas as normas de proteção de dados, os produtos, ao invés de nomeados com as respectivas marcas, foram numerados para diferenciar os nomes um em relação ao outro. Todos os produtos e dimensões podem ser observados na Tabela 1.

Após a definição dos produtos, deve-se considerar e calcular para que os compartimentos sejam de tamanho suficiente para o armazenamento de todos os itens. Desta maneira e baseando-se na Tabela 1, o produto com menor altura possui 45mm, já o maior produto possui

Tabela 1 – Descrição dos produtos contendo altura, largura e comprimento

ID	Nome	Altura	Largura	Comprimento
1	Perfume 1	120	90	40
2	Hidratante 1	200	80	50
3	Shampoo 1	210	70	40
4	Desodorante 1	210	45	45
5	Mamadeira 1	160	70	70
6	Lenço Umedecido 1	150	180	120
7	Pomada Assaduras 1	45	150	30
8	Vitamina D Frasco	100	45	45
9	Nitazoxanida	125	80	65
10	Aerolyn Spray	100	60	40
11	Montelucaste de sódio	100	70	80
12	Perfume 2	120	90	40
13	Hidratante 2	200	80	50
14	Shampoo 2	210	70	40
15	Desodorante 2	210	45	45
16	Mamadeira 2	160	70	70
17	Lenço Umedecido 2	150	180	120
18	Pomada Assaduras 2	45	150	30
19	Perfume 3	120	90	40
20	Hidratante 3	200	80	50
21	Shampoo 3	210	70	40
22	Desodorante 3	210	45	45
23	Mamadeira 3	160	70	70
24	Lenço Umedecido 3	150	180	120
25	Pomada Assaduras 3	45	150	30
26	Perfume 4	120	90	40
27	Hidratante 4	200	80	50
28	Shampoo 4	210	70	40
29	Desodorante 4	210	45	45
30	Lenço Umedecido 4	150	180	120

Fonte: Autoria própria (2024).

210 mm. Isto implica em que a altura do compartimento seja maior que 210 mm, neste caso optou-se por utilizar 220 mm de altura. Para a largura do compartimento, também é necessário analisar as dimensões dos produtos utilizados, sendo o menor produto 45 mm e o maior produto 180 mm. Considerando o tamanho do maior produto, optou-se por utilizar 190 mm de largura. Para o comprimento do compartimento, analisando os produtos tem-se que o menor possui comprimento de 30 mm e o maior comprimento de 120 mm. Dessa maneira a profundidade escolhida foi de 130 mm.

O volume e a frequência de seleção de cada produto pode ser observado na Tabela 2. Com todos esses dados indicados, pode-se iniciar a implementação do algoritmo MIP, declarando todas as variáveis definidas previamente.

Tabela 2 – Descrição dos produtos contendo volume e frequência

ID	Nome	Volume (mm3)	Frequência (%)
1	Perfume 1	432000	1,38
2	Hidratante 1	800000	5,76
3	Shampoo 1	588000	4,23
4	Desodorante 1	425250	4,89
5	Mamadeira 1	784000	5,11
6	Lenço Umedecido 1	3240000	2,19
7	Pomada Assaduras 1	202500	7,43
8	Vitamina D Frasco	202500	7,18
9	Nitazoxanida	650000	6,1
10	Aerolyn Spray	240000	1,68
11	Montelucaste de sódio	560000	4,84
12	Perfume 2	432000	0,23
13	Hidratante 2	800000	2,09
14	Shampoo 2	588000	0,56
15	Desodorante 2	425250	1,14
16	Mamadeira 2	784000	5,98
17	Lenço Umedecido 2	3240000	2,19
18	Pomada Assaduras 2	202500	5,36
19	Perfume 3	432000	2,55
20	Hidratante 3	800000	4,69
21	Shampoo 3	588000	1,93
22	Desodorante 3	425250	1,45
23	Mamadeira 3	784000	0,75
24	Lenço Umedecido 3	3240000	3,37
25	Pomada Assaduras 3	202500	2,78
26	Perfume 4	432000	1,49
27	Hidratante 4	800000	3,75
28	Shampoo 4	588000	1,43
29	Desodorante 4	425250	4,56
30	Lenço Umedecido 4	3240000	2,91

Fonte: Autoria própria (2024).

5.1 Implementação do algoritmo

O algoritmo é dividido em algumas etapas necessárias para que solucione o problema. Inicia-se com a declaração das variáveis utilizadas no processo. Para este problema foram utilizadas variáveis de entrada como: volume e frequência. Os valores foram escritos conforme o Algoritmo 2.

Deve-se garantir que volume e frequência possuirão o mesmo tamanho de vetor, pois é o que permite realizar a correspondência de volume e frequência por posição, indicando que o volume na posição 1 do vetor possuirá o valor de frequência de seleção na posição 1 do vetor de frequência.

Após as variáveis de entrada definidas, são declaradas as variáveis do sistema, elas auxiliam para manipular e atender as restrições e também à função objetivo do algoritmo. Para

Algoritmo 2 – Declaração das variáveis.

```
{Resolver o problema de multiplas mochilas usando o MIP Solver.}
from ortools.linear_solver import pywraplp

def main():
data = {}
data['volume'] = [...
432000, 800000, 588000, 425250, 784000, 3240000, 202500, 202500, ...
650000, 240000, 560000, 432000, 800000, 588000, 425250, 784000, ...
3240000, 202500, 432000, 800000, 588000, 425250, 784000, 3240000, ...
202500, 432000, 800000, 588000, 425250, 3240000]
data['frequencia'] = [...
1.38, 5.76, 4.23, 4.89, 5.11, 2.19, 7.43, 7.18, 6.1, 1.68, ...
4.84, 0.23, 2.09, 0.56, 1.14, 5.98, 2.19, 5.36, 2.55, 4.69, ...
1.93, 1.45, 0.75, 3.37, 2.78, 1.49, 3.75, 1.43, 4.56, 2.91]
```

Fonte: Autoria própria (2024).

o número total de itens do vetor a ser criado, é verificado o comprimento do vetor dos volumes. Nesta etapa também é definido o número de compartimentos e seu volume máximo. As variáveis podem ser observadas no Algoritmo 3.

Algoritmo 3 – Detalhes do Compartimento.

```
{Compartimentos}
assert len(data['volume']) == len(data['frequencia'])
data['num_itens'] == len(data['volume'])
data['total_itens'] = range(data['num_itens'])
data['bin_capacidade'] = [...
5434000, 5434000, 5434000, 5434000, 5434000, 5434000, ...
5434000, 5434000, 5434000, 5434000, 5434000, 5434000]
data['num_bins'] = len(data['bin_capacidade'])
data['total_bins'] = range(data['num_bins'])
```

Fonte: Autoria própria (2024).

Com as declarações de variáveis realizadas, deve-se escolher o tipo de *solver* utilizado para solucionar o problema. A partir desta seção do código, as duas soluções tem algumas diferenças no que diz respeito à declaração do *solver* e do modelo, isto pode ser observado no Algoritmo 4 e no Algoritmo 5.

Após a definição do *solver* com as variáveis utilizadas no sistema, o algoritmo segue para as restrições, que também possuem diferenças no modo de como são declaradas em cada modelo, conforme o Algoritmo 6 e o Algoritmo 7.

As restrições neste projeto são as seguintes:

- Cada item pode ser colocado em, no máximo, um agrupamento. Essa restrição é definida pela exigência da soma de $x[i, j]$ em todos os agrupamentos por classes j para ser menor ou igual a 1;

Algoritmo 4 – Criação do Solucionador para MIP *Solver*.

```
{Criar o MIP Solver com o SCIP/GLOP backend.}
solver = pywraplp.Solver.CreateSolver('SCIP')
se solver is None: então
    imprime ('Solucionador MIP não disponível)
finaliza se
{Variáveis}
{x[i, b] = 1 se item i está no bin b.}
x = {}
para i in data['total_itens'] : faça
    para b in data['total_bins'] : faça
        x{i,b} = solver.BoolVar(f"x_{i}_{b}")
    finaliza para
finaliza para
```

Fonte: Autoria própria (2024).

Algoritmo 5 – Criação do Solucionador para CP-SAT.

```
{Criar o modelo para o Solver com o CP-SAT backend.}
model = cp_model.CpModel() {Variáveis}
{x[i, b] = 1 se item i está no bin b.}
x = {}
para i in data['total_itens'] : faça
    para b in data['total_bins'] : faça
        x{i,b} = model.NewBoolVar(f"x_{i}_{b}")
    finaliza para
finaliza para
```

Fonte: Autoria própria (2024).

Algoritmo 6 – Definição das Restrições para MIP *Solver*.

```
{Restriçoes.}
{Cada item está alocado a no máximo um bin.}
para i in data['total_itens'] : faça
    solver.Add(sum(x[i,b] for b in data['total_bins']) <= 1)
finaliza para

{A quantidade alocada em cada bin não pode exceder sua capacidade.}
para b in data['total_bins'] : faça
    solver.Add(sum(x[i,b] * data['volume'][i] for i in ...
    data['total_itens']) <= data['bin_capacidade'][b])
finaliza para
```

Fonte: Autoria própria (2024).

- O peso total em cada recipiente não pode exceder sua capacidade. Para definir essa restrição, é necessário que a soma dos pesos dos itens colocados no agrupamento j seja menor ou igual à capacidade do agrupamento.

Com as restrições estabelecidas, parte-se para a função objetivo. Neste caso ela irá maximizar a frequência dos objetos selecionados, onde o código para os objetivos com o MIP *solver*

Algoritmo 7 – Definição das Restrições para CP-SAT.

```
{Restriçoes.}
{Cada item está alocado a no máximo um bin.}
para i in data['total_itens'] : faça
    model.AddAtMostOne(x[i,b] for b in data['total_bins'])
finaliza para

{A quantidade alocada em cada bin não pode exceder sua capacidade.}
para b in data['total_bins'] : faça
    model.Add(sum(x[i,b] * data['volume'][i] for i in ···
    data['tota_itens']) <= data['bin_capacidade'][b])
finaliza para
```

Fonte: Autoria própria (2024).

é representado no Algoritmo 8 e o código dos objetivos com o *solver* CP-SAT é representado no Algoritmo 9.

Algoritmo 8 – Objetivo da solução para MIP *Solver*.

```
{Objetivo.}
{Maximizar a frequência total dos itens alocados.}
objective = solver.Objective()
para i in data['total_itens'] : faça
    para b in data['total_bins'] : faça
        objective.SetCoeficient(x[i,b],data['frequencia'][i])
    finaliza para
finaliza para
objective.SetMaximization()
```

Fonte: Autoria própria (2024).

Algoritmo 9 – Objetivo da solução para CP-SAT.

```
{Objetivo.}
{Maximizar a frequência total dos itens alocados.}
objective = []
para i in data['total_itens'] : faça
    para b in data['total_bins'] : faça
        objective.append(cp_model.LinearExpr.Term(x[i,b],data['frequencia'][i]))
    finaliza para
finaliza para
model.Maximize(cp_model.LinearExpr.Sum(objective))
```

Fonte: Autoria própria (2024).

Observa-se no Algoritmo 8 que $x\,[i,j]\,,data\,[\text{"}frequencia\text{"}]\,[i]$ adicionará o valor do item i ao objetivo se ele for colocado no compartimento j. Se i não for colocado em nenhum compartimento, o valor dele não contribui para o objetivo.

Finalmente, a solução do problema para ambos os algoritmos pode ser apresentada de modo a imprimir os resultados em um terminal. Ambos os códigos buscam uma solução

ótima do problema de acordo com o pacote de como foram escritos, sendo a finalidade dos dois algoritmos a mesma.

Algoritmo 10 – Definição da resposta da solução para MIP *Solver*.

```
status = solver.Solve()

se status == pywraplp.Solver.OPTIMAL : então
    imprime (f'Total de Frequencia Preenchido: objective.Value()')
    total_volume = 0
    para b in data['total_bins'] : faça
        imprime ('Bin ' b + 1)
        bin_volume = 0
        bin_frequencia = 0
        para i in data['total_itens'] faça
            se x[i, b].solution_value() > 0 então
                imprime (f"Item i + 1 volume: data['volume'][i] frequencia: data['frequencia'][i]")
                bin_volume+ = data['volume'][i]
                bin_frequencia+ = data['frequencia'][i]
            finaliza se
        finaliza para
        imprime (f'Volume do Bin preenchido: bin_volume')
        imprime (f'Maior Frequencia do Bin preenchido: bin_frequencia')
        total_volume+ = bin_volume
    finaliza para
    imprime (f'Total de Volume Preenchido: total_volume')
senão,
    imprime ("O problema não possui uma solução ótima.")
finaliza se
```

Fonte: Autoria própria (2024).

Conforme o Algoritmo 10 e o Algoritmo 11, para cada conjunto, o códigos exibem os itens colocados nesse grupo, assim como o valor total e o peso do conjunto. Também é exibido o valor total e o peso dos itens no pacote.

5.2 Implementação do sistema

Os resultados da aplicação de ambos os algoritmos são apresentados na sequência, mostrando a alocação dos produtos e os *bins* nos quais foram inseridos.

Pode-se observar na Figura 10 o resultado apresentado do algoritmo de otimização linear SCIP para o problema. Já na Figura 11 é apresentado o resultado do algoritmo de otimização linear GLOP para o problema. Por fim, tem-se também o resultado do algoritmo de otimização linear CP-SAT apresentado na Figura 12. Consegue-se verificar que somente nas técnicas de otimização SCIP e CP-SAT são respeitadas todas as restrições definidas para o sistema, como por exemplo na questão do volume dos compartimentos, onde nenhum compartimento ficou com volume de produtos maior que a capacidade do mesmo. Observou-se que nenhum problema ocorreu durante a implementação dos códigos como erros e alertas e tam-

Algoritmo 11 – Definição da resposta da solução para CP-SAT.

```
solver = cp_model.CpSolve()
status = solver.Solve(model)

se status == cp_model.OPTIMAL : então
    imprime (f'Total de Frequencia Preenchido: solver.ObjectiveValue()')
    total_volume = 0
    para b in data['total_bins'] : faça
        imprime ('Bin ' b + 1)
        bin_volume = 0
        bin_frequencia = 0
        para i in data['total_itens'] faça
            se solver.Value(x[i, b]) > 0 então
                imprime (f"Item i + 1 volume: data['volume'][i] frequencia: data['frequencia'][i]")
                bin_volume+ = data['volume'][i]
                bin_frequencia+ = data['frequencia'][i]
            finaliza se
        finaliza para
        imprime (f'Volume do Bin preenchido: bin_volume')
        imprime (f'Maior Frequencia do Bin preenchido: bin_frequencia')
        total_volume+ = bin_volume
    finaliza para
    imprime (f'Total de Volume Preenchido: total_volume')
senão,
    imprime ("O problema não possui uma solução ótima.")
finaliza se
```

Fonte: Autoria própria (2024).

bém que os programas alocaram corretamente todos os produtos, sem deixar nenhum de fora da armazenagem. Já para o *solver* GLOP, o algoritmo não respeitou as restrições impostas pelo sistema, onde, observando o resultado, um mesmo item foi alocado mais de uma vez em compartimentos diferentes e dois compartimentos tiveram sua capacidade de armazenamento excedida, não respeitando outra restrição imposta pelo sistema.

5.3 Discussões

Com relação aos resultados obtidos, pode-se observar que todos os produtos estavam corretamente indicados com relação ao volume e à frequência e seu índice. As posições também destacadas como compartimentos eram 12 iniciais com capacidade de armazenamento igual entre elas.

Avaliando a Figura 13, pode-se observar que o *solver* SCIP realiza uma distribuição dos produtos preenchendo os *bins* de modo a alocar todos os itens corretamente. Considerando o critério de otimização como deslocamento do operador, o *solver* consegue chegar em um resultado satisfatório pois concentra os produtos sempre nos compartimentos iniciais, deste

Figura 10 – Resultado do Algoritmo - *Solver* SCIP

```
Total de Frequência Preenchido: 100.0%
Total de Volume Preenchido: 26323500

Bin 1
Item 1 volume: 432000 frequência: 1.38
Item 2 volume: 800000 frequência: 5.76
Item 3 volume: 588000 frequência: 4.23
Item 4 volume: 425250 frequência: 4.89
Item 5 volume: 784000 frequência: 5.11
Item 7 volume: 202500 frequência: 7.43
Item 8 volume: 202500 frequência: 7.18
Item 9 volume: 650000 frequência: 6.1
Item 10 volume: 240000 frequência: 1.68
Item 11 volume: 560000 frequência: 4.84
Item 12 volume: 432000 frequência: 0.23
Volume do Bin preenchido: 5316250
Maior Frequência do Bin preenchido: 48.83

Bin 2
Item 6 volume: 3240000 frequência: 2.19
Item 13 volume: 800000 frequência: 2.09
Item 14 volume: 588000 frequência: 0.56
Item 15 volume: 425250 frequência: 1.14
Item 18 volume: 202500 frequência: 5.36
Volume do Bin preenchido: 5255750
Maior Frequência do Bin preenchido: 11.34

Bin 3
Item 16 volume: 784000 frequência: 5.98
Item 17 volume: 3240000 frequência: 2.19
Item 19 volume: 432000 frequência: 2.55
Item 20 volume: 800000 frequência: 4.69
Volume do Bin preenchido: 5256000
Maior Frequência do Bin preenchido: 15.41

Bin 4
Item 21 volume: 588000 frequência: 1.93
Item 22 volume: 425250 frequência: 1.45
Item 23 volume: 784000 frequência: 0.75
Item 24 volume: 3240000 frequência: 3.37
Item 25 volume: 202500 frequência: 2.78
Volume do Bin preenchido: 5239750
Maior Frequência do Bin preenchido: 10.28

Problem solved in 40 milliseconds
Problem solved in 260 iterations

Bin 5
Item 26 volume: 432000 frequência: 1.49
Item 27 volume: 800000 frequência: 3.75
Item 28 volume: 588000 frequência: 1.43
Item 29 volume: 425250 frequência: 4.56
Volume do Bin preenchido: 2245250
Maior Frequência do Bin preenchido: 11.23

Bin 6
Item 30 volume: 3240000 frequência: 2.91
Volume do Bin preenchido: 3240000
Maior Frequência do Bin preenchido: 2.91

Bin 7
Volume do Bin preenchido: 0
Maior Frequência do Bin preenchido: 0

Bin 8
Volume do Bin preenchido: 0
Maior Frequência do Bin preenchido: 0

Bin 9
Volume do Bin preenchido: 0
Maior Frequência do Bin preenchido: 0

Bin 10
Volume do Bin preenchido: 0
Maior Frequência do Bin preenchido: 0

Bin 11
Volume do Bin preenchido: 0
Maior Frequência do Bin preenchido: 0

Bin 12
Volume do Bin preenchido: 0
Maior Frequência do Bin preenchido: 0
```

Fonte: Autoria própria (2024).

modo atende aos critérios de restrições estabelecidos, sendo seu resultado considerado válido para a análise.

Levando em consideração os resultados da Figura 14 como comparação, pode-se observar que o *solver* GLOP realiza uma distribuição dos produtos preenchendo os *bins* de modo a alocar repetidamente os itens em diversos compartimentos. Além disto, o algoritmo não respeita as capacidades dos *bins* permitindo alocar mais produtos do que o permitido. Considerando a distribuição realizada, ela se justificaria se a estação de *picking* estiver dimensionada para que mais operadores possam trabalhar ao mesmo tempo. O algoritmo não atende aos critérios de restrições estabelecidos, sendo seu resultado não considerado válido para a análise. Ainda sobre o GLOP, o *solver* se demonstra eficaz quando atua em problemas com variáveis contínuas, não sendo especificado para programação inteira ou otimização combinatória. A utilização dele

Figura 11 – Resultado do Algoritmo - *Solver* GLOP

```
Total de Frequência Preenchido: 145.3%
Total de Volume Preenchido: 38585000

Bin 1
Item 1 volume: 432000 frequência: 1.38
Item 16 volume: 784000 frequência: 5.98
Item 21 volume: 588000 frequência: 1.93
Item 25 volume: 202500 frequência: 2.78
Item 28 volume: 588000 frequência: 1.43
Volume do Bin preenchido: 2594500
Maior Frequência do Bin preenchido: 13.5

Bin 2
Item 7 volume: 202500 frequência: 7.43
Item 11 volume: 560000 frequência: 4.84
Item 27 volume: 800000 frequência: 3.75
Volume do Bin preenchido: 1562500
Maior Frequência do Bin preenchido: 16.02

Bin 3
Item 3 volume: 588000 frequência: 4.23
Item 5 volume: 784000 frequência: 5.11
Item 9 volume: 650000 frequência: 6.1
Item 12 volume: 432000 frequência: 0.23
Item 19 volume: 432000 frequência: 2.55
Item 20 volume: 800000 frequência: 4.69
Item 27 volume: 800000 frequência: 3.75
Item 29 volume: 425250 frequência: 4.56
Volume do Bin preenchido: 4911250
Maior Frequência do Bin preenchido: 31.22

Bin 4
Item 6 volume: 3240000 frequência: 2.19
Item 13 volume: 800000 frequência: 2.09
Item 24 volume: 3240000 frequência: 3.37
Volume do Bin preenchido: 7280000
Maior Frequência do Bin preenchido:
7.6499999999999995

Bin 5
Item 16 volume: 784000 frequência: 5.98
Item 17 volume: 3240000 frequência: 2.19
Item 18 volume: 202500 frequência: 5.36
Item 30 volume: 3240000 frequência: 2.91
Volume do Bin preenchido: 7466500
Maior Frequência do Bin preenchido: 16.44

Problem solved in 10 milliseconds
Problem solved in 60 iterations

Bin 6
Item 17 volume: 3240000 frequência: 2.19
Item 23 volume: 784000 frequência: 0.75
Volume do Bin preenchido: 4024000
Maior Frequência do Bin preenchido: 2.94

Bin 7
Item 2 volume: 800000 frequência: 5.76
Item 10 volume: 240000 frequência: 1.68
Item 23 volume: 784000 frequência: 0.75
Volume do Bin preenchido: 1824000
Maior Frequência do Bin preenchido: 8.19

Bin 8
Item 4 volume: 425250 frequência: 4.89
Item 14 volume: 588000 frequência: 0.56
Item 26 volume: 432000 frequência: 1.49
Volume do Bin preenchido: 1445250
Maior Frequência do Bin preenchido:
6.9399999999999995

Bin 9
Item 13 volume: 800000 frequência: 2.09
Volume do Bin preenchido: 800000
Maior Frequência do Bin preenchido: 2.09

Bin 10
Item 5 volume: 784000 frequência: 5.11
Item 20 volume: 800000 frequência: 4.69
Item 22 volume: 425250 frequência: 1.45
Volume do Bin preenchido: 2009250
Maior Frequência do Bin preenchido: 11.25

Bin 11
Item 8 volume: 202500 frequência: 7.18
Volume do Bin preenchido: 202500
Maior Frequência do Bin preenchido: 7.18

Bin 12
Item 2 volume: 800000 frequência: 5.76
Item 6 volume: 3240000 frequência: 2.19
Item 15 volume: 425250 frequência: 1.14
Volume do Bin preenchido: 4465250
Maior Frequência do Bin preenchido: 9.09
```

Fonte: Autoria própria (2024).

no trabalho tem um objetivo maior de comparação e avaliação da funcionalidade quando apresentado à um problema de MIP.

Já na Figura 15, pode-se observar que o *solver* CP-SAT realiza uma distribuição dos produtos preenchendo os *bins* de modo a alocar todos os itens corretamente. Considerando o critério de otimização como deslocamento do operador, este *solver* é pior quando comparado ao SCIP pois realiza um espalhamento maior dos produtos nos *bins*. Porém caso seja necessária a adição de mais de um operador para a realização dos pedidos de uma mesma estação, esta distribuição mais espalhada permite o acesso aos itens sem a necessidade de espera ou conflito

Figura 12 – Resultado do Algoritmo - *Solver* CP-SAT

```
Total de Frequência Preenchido: 100.0%
Total de Volume Preenchido: 26323500

Bin 1
Item 7 volume: 202500 frequência: 7.43
Item 8 volume: 202500 frequência: 7.18
Item 30 volume: 3240000 frequência: 2.91
Volume do Bin preenchido: 3645000
Maior Frequência do Bin preenchido: 17.52

Bin 2
Item 6 volume: 3240000 frequência: 2.19
Volume do Bin preenchido: 3240000
Maior Frequência do Bin preenchido: 2.19

Bin 3
Item 1 volume: 432000 frequência: 1.38
Item 4 volume: 425250 frequência: 4.89
Volume do Bin preenchido: 857250
Maior Frequência do Bin preenchido: 6.27

Bin 4
Item 2 volume: 800000 frequência: 5.76
Item 3 volume: 588000 frequência: 4.23
Item 9 volume: 650000 frequência: 6.1
Item 11 volume: 560000 frequência: 4.84
Item 12 volume: 432000 frequência: 0.23
Item 14 volume: 588000 frequência: 0.56
Item 19 volume: 432000 frequência: 2.55
Item 22 volume: 425250 frequência: 1.45
Item 23 volume: 784000 frequência: 0.75
Volume do Bin preenchido: 5259250
Maior Frequência do Bin preenchido: 26.47

Bin 5
Volume do Bin preenchido: 0
Maior Frequência do Bin preenchido: 0

Bin 6
Item 5 volume: 784000 frequência: 5.11
Volume do Bin preenchido: 784000
Maior Frequência do Bin preenchido: 5.11

Problem solved in 50 milliseconds
Problem solved in    iterations

Bin 7
Volume do Bin preenchido: 0
Maior Frequência do Bin preenchido: 0

Bin 8
Item 13 volume: 800000 frequência: 2.09
Item 15 volume: 425250 frequência: 1.14
Item 16 volume: 784000 frequência: 5.98
Item 26 volume: 432000 frequência: 1.49
Item 29 volume: 425250 frequência: 4.56
Volume do Bin preenchido: 2866500
Maior Frequência do Bin preenchido: 15.26

Bin 9
Item 20 volume: 800000 frequência: 4.69
Item 28 volume: 588000 frequência: 1.43
Volume do Bin preenchido: 1388000
Maior Frequência do Bin preenchido: 6.12

Bin 10
Item 10 volume: 240000 frequência: 1.68
Item 18 volume: 202500 frequência: 5.36
Item 21 volume: 588000 frequência: 1.93
Item 25 volume: 202500 frequência: 2.78
Item 27 volume: 800000 frequência: 3.75
Volume do Bin preenchido: 2033000
Maior Frequência do Bin preenchido: 15.5

Bin 11
Item 24 volume: 3240000 frequência: 3.37
Volume do Bin preenchido: 3240000
Maior Frequência do Bin preenchido: 3.37

Bin 12
Item 17 volume: 3240000 frequência: 2.19
Volume do Bin preenchido: 3240000
Maior Frequência do Bin preenchido: 2.19
```

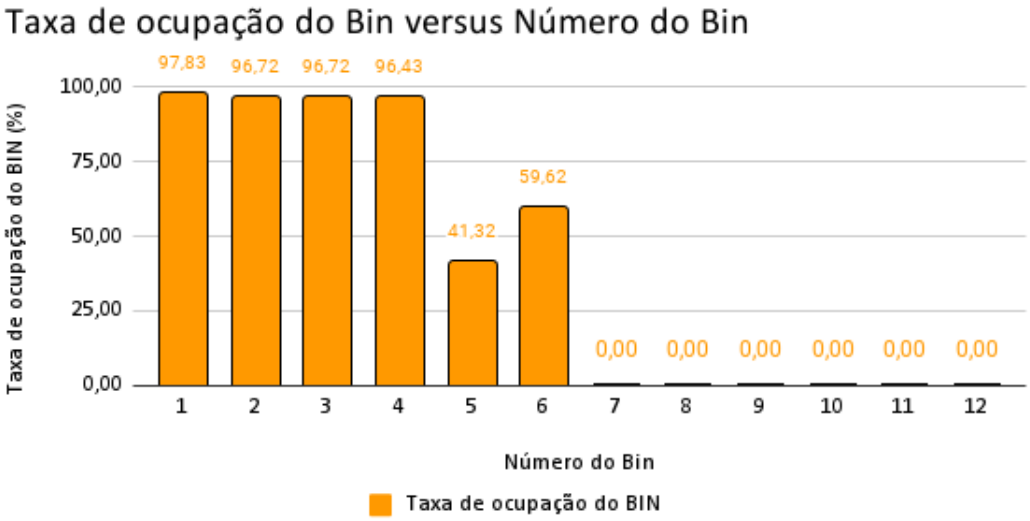
Fonte: Autoria própria (2024).

com o outro operador. O algoritmo atende aos critérios de restrições estabelecidos, sendo seu resultado considerado válido para a análise.

Outra comparação que pode ser realizada com os dados retirados deste trabalho, é a distribuição das frequências (custo) alocadas em cada *bin*. O objetivo de cada *solver* é a otimização do custo.

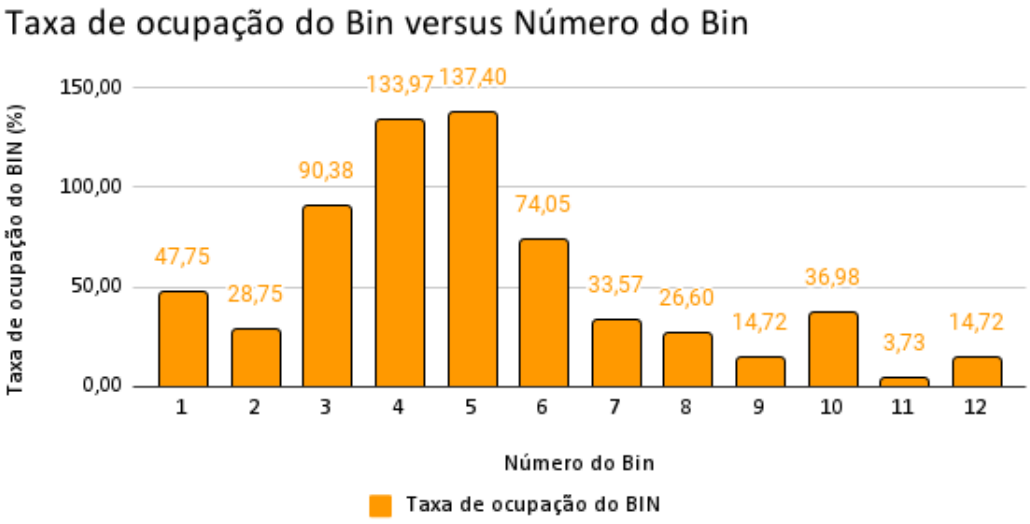
Os algoritmos propostos buscam otimizar a frequência porém não podem ultrapassar a capacidade indicada em cada compartimento. Como pode-se verificar na Figura 16, o algoritmo que utiliza o solucionador SCIP busca preencher com o máximo de frequência possível o primeiro *bin* e nos demais que restam, busca otimizar ainda a frequência mas sem ultrapassar a capacidade pois acaba deixando para o fim da separação os itens de maior volume com menor encaixe entre outros produtos.

Figura 13 – Taxa de ocupação dos *bins* - *Solver* SCIP



Fonte: Autoria própria (2024).

Figura 14 – Taxa de ocupação dos *bins* - *Solver* GLOP

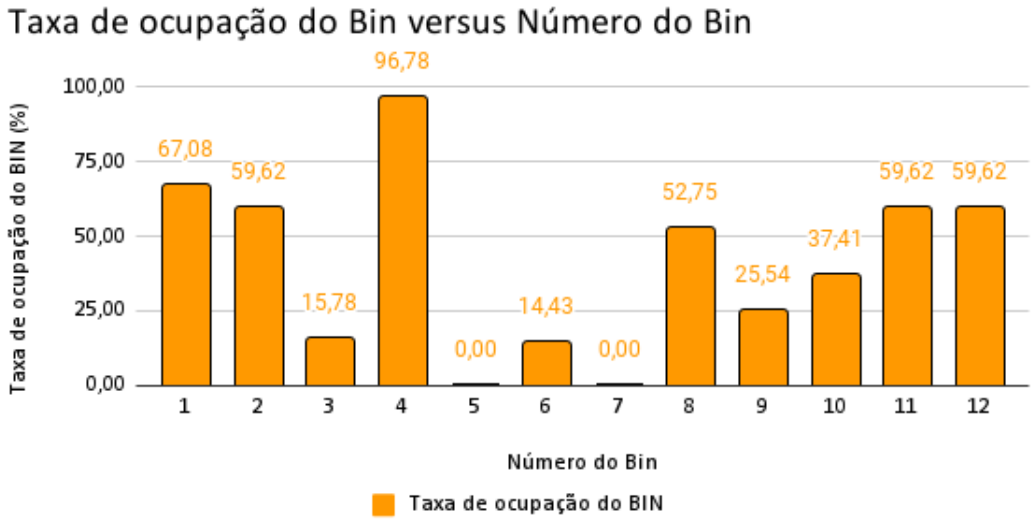


Fonte: Autoria própria (2024).

Para a Figura 17, realizou-se somente a elaboração do gráfico porém o mesmo não é válido como forma de analisar o comportamento do *solver* GLOP. Como houve a repetição de diversos itens em compartimentos diferentes, a frequência ultrapassou os $100{,}00\%$ chegando a um somatório de $145{,}30\%$, tornando inviável a comparação deste método com outro.

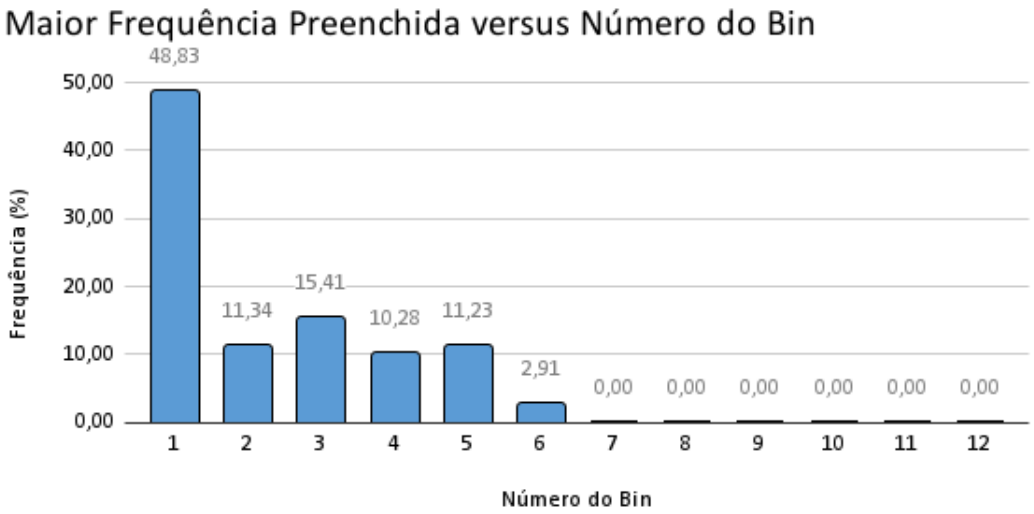
Finalizando esta análise, pode-se observar na Figura 18, com o algoritmo CP-SAT que as distribuições de frequência são mais espalhadas porém só superam a casa dos $10{,}00\%$ em 4 compartimentos. Estes compartimentos somados correspondem a $74{,}75\%$ de todos o produtos alocados na estrutura de armazenagem. Considerando a frequência como o custo, o algoritmo conseguiu atender o que necessitava pois otimizou de modo a deixar a menor parte das frequências espalhadas em *bins* mais distantes do operador.

Figura 15 – Taxa de ocupação dos *bins* - *Solver* CP-SAT



Fonte: Autoria própria (2024).

Figura 16 – Maior Frequência Preenchida nos *bins* - *Solver* SCIP



Fonte: Autoria própria (2024).

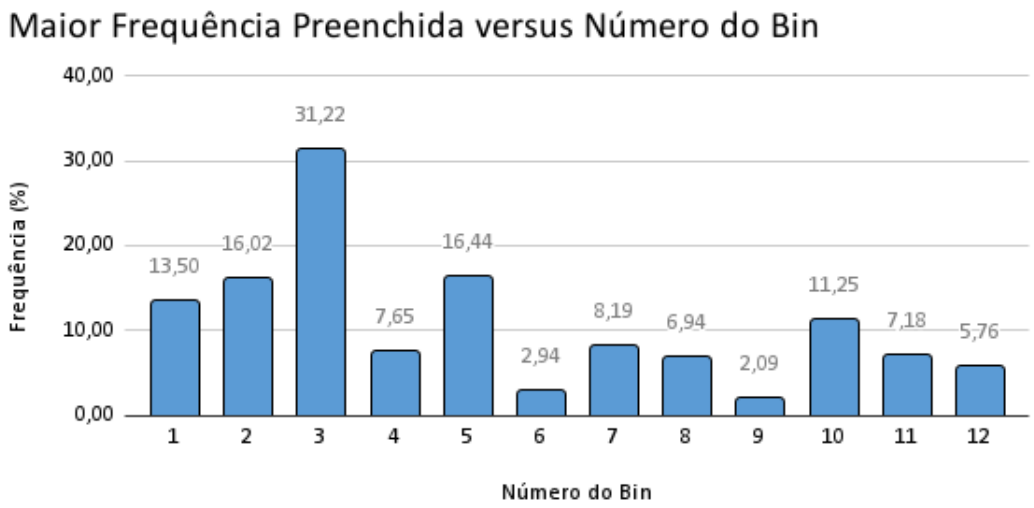
5.3.1 Pontos positivos e negativos

Nesta subseção são apresentados alguns pontos positivos e negativos:

Pontos positivos:

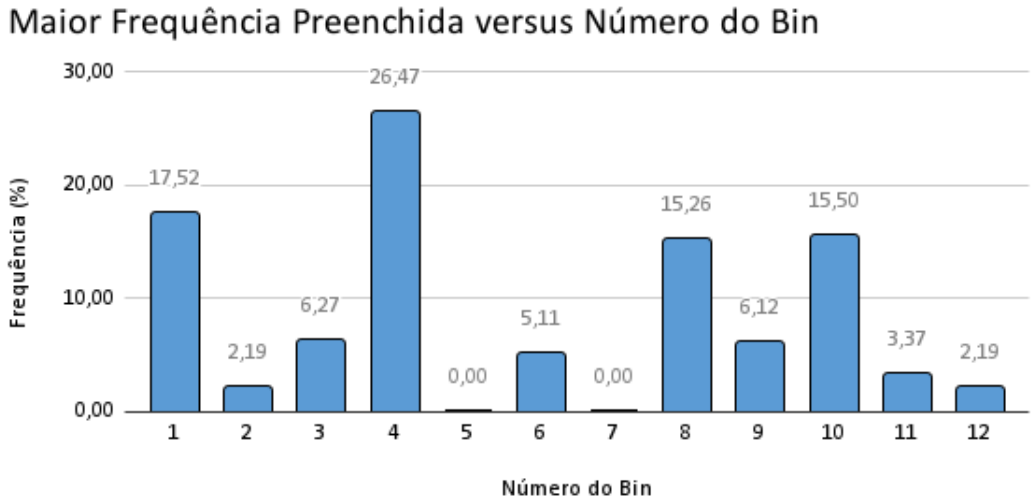
- Eficiência na resolução de problemas de otimização combinatória (incluindo MIP), onde os *solvers* de acordo com a sua finalidade são capazes de lidar com uma grande variedade de objetivos e restrições;
- Os *solvers* utilizados possuem código aberto e altamente configurável;

Figura 17 – Maior Frequência Preenchida nos *bins* - *Solver* GLOP



Fonte: Autoria própria (2024).

Figura 18 – Maior Frequência Preenchida nos *bins* - *Solver* CP-SAT



Fonte: Autoria própria (2024).

- Permite a integração do *solver* com o sistema de gerenciamento do armazém, agilizando o processo de distribuição dos produtos nos *Flow-Rack*.

Pontos negativos:

- Trabalham apenas com inteiros, o que significa que o problema deve ser definido utilizando somente inteiros;
- Para problemas muito grandes ou altamente complexos, o tempo de computação pode ser significativo, mesmo com *solvers* eficientes.

6 CONCLUSÃO

Este trabalho apresentou a otimização de alocação de itens em uma estrutura de armazenagem para produtos *Flow-rack* através da implementação de um algoritmo de otimização, conforme os Apêndices A e B. Para iniciar, foi montada uma base de dados de produtos e definidas as variáveis de acordo com os dados baseados em tamanhos de produtos reais de mercado de uma indústria farmacêutica. Os produtos foram escolhidos de modo a respeitar as dimensões dos compartimentos onde seriam alocados.

O trabalho abordou uma resolução através de otimização linear, que é uma área da otimização combinatória. A otimização linear apresentou boa resposta para o projeto por se tratar de uma estrutura linear que demanda menos recursos computacionais. Como justificativa, pode-se quantificar o tempo gasto pelo algoritmo para gerar a solução do código e quantas iterações foram necessárias para alcançar o resultado *OK*, isto é possível utilizando funções do próprio algoritmo. Isto irá variar de acordo com a estrutura em que este modelo será aplicado, quanto mais complexo o sistema mais recursos irá demandar para solucionar o mesmo. Outro ponto a ser citado é que matematicamente possui uma teoria bem desenvolvida e métodos sólidos e comprovados em sua resolução.

Dentre as soluções propostas, analisou-se o algoritmo com o solucionador SCIP, o solucionador GLOP, ambos projetados para problemas de programação linear e não linear mista que utilizam algoritmos de *branch-and-bound* para a solução dos problemas, empregando técnicas de relaxamentos de programação. Além disso, também foi utilizado o solucionador CP-SAT, que integra técnicas de programação de restrições com métodos de resolução de satisfazibilidade *booleana* (SAT). Este último solucionador representa um paradigma para resolver problemas de otimização combinatória, que envolve a especificação de restrições, condicionando a solução, permitindo que o solucionador busque um resultado que satisfaça todas as condições impostas.

Analisando os resultados obtidos, verifica-se que o algoritmo SCIP e CP-SAT conseguiram distribuir os produtos respeitando todos os critérios, restrições e função objetivo, de modo a maximizar a função, de acordo com seu *solver* e solucionar o problema. Com relação ao algoritmo do GLOP, ele não atendeu a nenhum dos critérios estabelecidos, sendo assim, não é recomendado para a resolução de problemas de MIP. Tendo em vista que o projeto também necessita olhar os aspectos ergonômicos, o *solver* CP-SAT peca em não realizar a distribuição mais concentrada dos itens levando em consideração a posição dos *bins* em relação à estrutura como um todo. Deste modo a distribuição dos itens de maneira ótima, concentra os itens e permite realizar a coleta sem maiores deslocamentos. Portanto, de acordo com a análise buscando melhores práticas ergonômicas, o SCIP permanece como melhor solução para este trabalho.

Podem ser apresentados alguns pontos positivos e negativos. Entre os pontos positivos, destaca-se a eficiência na resolução de problemas de otimização combinatória, incluindo MIP, onde os *solvers*, de acordo com sua finalidade, lidam com uma grande variedade de objetivos e restrições. Além disso, os *solvers* utilizados possuem código aberto e são altamente configu-

ráveis, permitindo sua integração com o sistema de gerenciamento do armazém e agilizando o processo de distribuição dos produtos nos *Flow-rack*. Por outro lado, entre os pontos negativos, ressalta-se que eles trabalham apenas com inteiros, exigindo que o problema seja definido utilizando somente inteiros. Ademais, para problemas muito grandes ou altamente complexos, o tempo de computação pode ser significativo, mesmo com *solvers* eficientes.

Para trabalhos futuros, pode-se avançar com a implementação prática do algoritmo em uma estação de trabalho real, isso pode envolver o uso de tecnologias como *pick-to-light* nos *Flow-racks* e robôs *pick-and-place* para identificar o produto e armazená-lo no compartimento correto. A pesquisa pode ser estendida para explorar a variação do tamanho dos *bins*. Isto engloba a identificação dos tamanhos de *bins* que seriam ergonomicamente melhores e que poderiam acomodar mais produtos e otimizar a frequência. Outro ponto importante seria de identificar centros de distribuição onde a implementação do algoritmo pode ter um impacto econômico considerável. Cada uma dessas áreas tem o potencial de contribuir significativamente para os processos de armazenamento em centros de distribuição.

REFERÊNCIAS

ACHTERBERG, T. Scip: solving constraint integer programs. **Mathematical Programming Computation**, Springer, v. 1, p. 1–41, 2009.

ALMEIDA, M. das N. *et al.* Modelagem e simulação de uma proposta de estoque flow rack para otimizar o layout do armazém de medicamentos e reduzir os desperdicíos. **Revista Produção Online**, v. 19, n. 3, p. 952–980, 2019.

AMARANTE, S. R. M. Utilizando o problema de múltiplas mochilas para modelar o problema de alocação de máquinas virtuais em computação nas nuvens. 2013.

ASSIS, R. d.; SAGAWA, J. K. Avaliação da implantação do sistema de gestão de armazém em uma empresa multinacional do ramo de acionamentos. **Gestão & Produção**, Universidade Federal de São Carlos, v. 25, n. 2, p. 370–383, Apr 2018. ISSN 0104-530X. Disponível em: https://doi.org/10.1590/0104-530X3315-18.

BALLOU, R. H. **Gerenciamento da Cadeia de Suprimentos-: Logística Empresarial**. [*S.l.*]: Bookman editora, 2009.

BANZATO, E. **WMS – Warehouse management system: Sistema de gerenciamento de armazéns. São Paulo**. [*S.l.*]: IMAN, 1998.

BRETAS, T. L. Problema da mochila inteira aplicado em decisões de compra de aparelhos de atividades desportivas: Modelo e aplicação. 2016.

CASTRO, D.; RAMOS, M.; COSTA, D. Estudo de tempos e movimentos no processo de flow rack em uma empresa de distribuição. **ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO**, v. 32, 2012.

CIA, H. E. M. **Avaliação de alternativas de layouts e fluxos internos visando melhorar a movimentação num centro de distribuição de bebidas**. [*S.l.*]: Fundação Vanzolini - CELOG, 2015.

CLEMENTE, Q. K. **Resolução de Problemas da Mochila Multicritério através de técnicas Metaheurísticas**. [*S.l.*]: Lisboa: Universidade Técnica de Lisboa, 2010.

COSTA, G. N. **A utilização da curva ABC como ferramenta de gerenciamento de estoque**. 2017. Dissertação (B.S. thesis) — Universidade Tecnológica Federal do Paraná, 2017.

COSTA, L. G. d.; VALÉRIO, T. P. **Melhoria de processos: um estudo aplicado em uma linha de produção de sistemas de armazenagem**. 2020. Dissertação (B.S. thesis) — Universidade Tecnológica Federal do Paraná, 2020.

CUNHA, J. O.; SIMONETTI, L.; LUCENA, A. Um algoritmo branch-and-cut para o problema da mochila quadrática 0-1. *In*: **46º Simpósio Brasileiro de Pesquisa Operacional**. [*S.l.*: *s.n.*], 2014. p. 3739–3750.

DIAS, M. A. P. **Administração de materiais: uma abordagem logística. São Paulo: Atlas**. [*S.l.*]: ISBN, 2010.

GROSSE, E. H.; GLOCK, C. H.; NEUMANN, W. P. Human factors in order picking: a content analysis of the literature. **International journal of production research**, Taylor & Francis, v. 55, n. 5, p. 1260–1276, 2017.

HAN, Y.-H.; ZHOU, C. Dynamic sequencing of jobs on conveyor systems for minimizing changeovers. **The International Journal of Advanced Manufacturing Technology**, Springer, v. 49, p. 1251–1259, 2010.

JANUÁRIO, K. D. Implementaçao da meta-heurıstica de busca em vizinhanças variáveis (vns) na soluçao do problema da mochila multidimensional. Fundação Universidade Federal de Mato Grosso do Sul, 2023.

JUNIOR, C. A. S. **Elaboração de um instrumento para a avaliação ergonômica preliminar em empresa do setor logístico**. 2023. 77 p. Dissertação (B.S. thesis) — Universidade Federal do Rio Grande do Sul, Porto Alegre, 2023.

LUILA, E. P. **Problema da Mochila Multicritério: Aspectos algorítmicos e Implementação Informática**. 2008. Dissertação (B.S. thesis) — Lisboa: Universidade Técnica de Lisboa, 2008.

MAAS, L. *et al.* Norma regulamentadora 17: considerações para sua revisão. **Human Factors in Design**, v. 9, n. 17, p. 137–162, 2020.

MARQUES, F. d. P.; ARENALES, M. N. O problema da mochila compartimentada e aplicações. **Pesquisa Operacional**, SciELO Brasil, v. 22, p. 285–304, 2002.

MARTELLO, S.; TOTH, P. **Knapsack problems: algorithms and computer implementations**. [*S.l.*]: John Wiley & Sons, Inc., 1990.

MARTINS, P. G.; LAUGENI, F. P. Administração da produção. Saraiva São Paulo, 2005.

MATTONE, R.; ADDUCI, L.; WOLF, A. Online scheduling algorithms for improving performance of pick-and-place operations on a moving conveyor belt. *In*: **Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)**. [*S.l.*: *s.n.*], 1998. v. 3, p. 2099–2105 vol.3.

NATÁRIO, D. J. T. **Melhoria da eficiência de processos de trabalho numa linha de montagem de componentes: articulação entre lean production e ergonomia**. oct 2017. 126 p. Tese (Doutorado) — Escola de Engenharia, Universidade do Minho, Braga, oct 2017.

NETO, F. L.; PEDROTTI, V.; HOSHINO, E. Matheurísticas para o problema de roteamento de veículos com prêmios. *In*: **Anais do VIII Encontro de Teoria da Computação**. Porto Alegre, RS, Brasil: SBC, 2023. p. 185–189. ISSN 2595-6116. Disponível em: https://sol.sbc.org.br/index.php/etc/article/view/24767.

NUNES, T. C. F. *et al.* **O processo de armazenagem de termolábeis: estudo em uma indústria farmacêutica no estado de Goiás**. [*S.l.*]: Insitituto Federal de Educação, Ciência e Tecnologia de Goiás, 2019.

OR-TOOLS, G. D. **Sobre as ferramentas OR**. 2023. Disponível em: https://developers.google.com/optimization/introduction?hl=pt-br. Acesso em: 20 de novembro de 2023.

PERRON, L.; DIDIER, F.; GAY, S. The CP-SAT-LP Solver. *In*: YAP, R. H. C. (Ed.). **29th International Conference on Principles and Practice of Constraint Programming (CP 2023)**. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. (Leibniz International Proceedings in Informatics (LIPIcs), v. 280), p. 3:1–3:2. ISBN 978-3-95977-300-3. ISSN 1868-8969. Disponível em: https://drops-dev.dagstuhl.de/entities/document/10.4230/LIPIcs.CP.2023.3.

PINTO, C. P. **A Rastreabilidade no Contexto da Gestão da Qualidade.** Abr 2016. 131 p. Dissertação (B.S. thesis) — Universidade Federal de Itajubá-Brasil, Itajubá, Abr 2016.

ROCHA, J. T. d. *et al.* A possibilidade de utilização do sistema wms para melhoria da gestão de armazenagem: o caso de uma empresa do ramo de construção da zona da mata mineira. **Brazilian Journal of Production Engineering**, v. 7, n. 5, p. 166–182, dez. 2021. Disponível em: https://periodicos.ufes.br/bjpe/article/view/36362.

ROSA, M. N. S. **Redesenho da cadeia de abastecimento de uma indústria farmacêutica–foco na alocação e na rede de distribuição**. oct 2023. 65 p. Tese (Doutorado) — Escola de Engenharia, Universidade do Minho, Braga, oct 2023.

SCHNELL, A.; HARTL, R. F. On the generalization of constraint programming and boolean satisfiability solving techniques to schedule a resource-constrained project consisting of multi-mode jobs. **Operations Research Perspectives**, Elsevier, v. 4, p. 1–11, 2017.

SISTEMAS, A. **Flow Rack: Águia Sistemas de Armazenagem**. 2023. Disponível em: https://www.aguiasistemas.com.br/br/solucoes/estruturas/flow-rack#picture-gallery. Acesso em: 03 de novembro de 2023.

VERÇOSA, N. J. **Modelo e algoritmos para seleção de sensores como serviço**. 2019. Dissertação (B.S. thesis) — Brasil, 2019.

VILLAR, A. d. M.; SILVA, L. M. F.; NOBREGA, M. M. **Planejamento, Programação e Controle da Produção**. Ponta Grossa/PR: Editora da UFPB, 2008. 364 p.

YAMASHITA, A.; GARCIA, L. Análise da aplicaçao do algoritmo col^onia de abelhas artificiais no problema da mochila multidimensional binária. **Trabalho de Conclusao de Curso. Universidade Federal de Mato Grosso do Sul, Faculdade de Computaçao**, 2023.

APÊNDICE A – Código Completo do Algoritmo de MIP para os Solucionadores SCIP e GLOP Escritos em Linguagem *Python*

```
"""Resolver o problema de multiplas mochilas usando o MIP Solver."""
from ortools.linear_solver import pywraplp

def main():
    data = {}
    data['volume'] = [
        432000, 800000, 588000, 425250, 784000, 3240000, 202500, 202500, 650000, 240000,
        560000, 432000, 800000, 588000, 425250, 784000, 3240000, 202500, 432000, 800000,
        588000, 425250, 784000, 3240000, 202500, 432000, 800000, 588000, 425250, 3240000]
    data['frequencia'] = [
        1.38, 5.76, 4.23, 4.89, 5.11, 2.19, 7.43, 7.18, 6.1, 1.68,
        4.84, 0.23, 2.09, 0.56, 1.14, 5.98, 2.19, 5.36, 2.55, 4.69,
        1.93, 1.45, 0.75, 3.37, 2.78, 1.49, 3.75, 1.43, 4.56, 2.91]

    assert len(data['volume']) == len(data['frequencia'])
    data['num_itens'] = len(data['volume'])
    data['total_itens'] = range(data['num_itens'])
    data['bin_capacidade'] = [5434000, 5434000, 5434000, 5434000, 5434000, 5434000,
                              5434000, 5434000, 5434000, 5434000, 5434000, 5434000]
    data['num_bins'] = len(data['bin_capacidade'])
    data['total_bins'] = range(data['num_bins'])

    # Criar o MIP Solver com o SCIP/GLOP backend.
    solver = pywraplp.Solver.CreateSolver('SCIP')
    if solver is None:
        print('Solucionador MIP não disponível.')
        return

    # Variaveis.
    # x[i, b] = 1 se item i está no bin b.
    x = {}
    for i in data['total_itens']:
        for b in data['total_bins']:
            x[i, b] = solver.BoolVar(f'x_{i}_{b}')

    # Restriçoes.
    # Cada item está alocado a no máximo um bin.
    for i in data['total_itens']:
        solver.Add(sum(x[i, b] for b in data['total_bins']) <= 1)

    # A quantidade alocada em cada bin não pode exceder sua capacidade.
    for b in data['total_bins']:
        solver.Add(
            sum(x[i, b] * data['volume'][i] for i in data['total_itens'])
            <= data['bin_capacidade'][b])

    # Objetivo.
    # Maximizar a frequencia total dos itens alocados.
    objective = solver.Objective()
    for i in data['total_itens']:
        for b in data['total_bins']:
            objective.SetCoefficient(x[i, b], data['frequencia'][i])
    objective.SetMaximization()

    status = solver.Solve()

```

```
57      if status == pywraplp.Solver.OPTIMAL:
58          print(f'Total de Frequencia Preenchido: {objective.Value()}')
59          total_volume = 0
60          for b in data['total_bins']:
61              print(f'Bin {b+1}')
62              bin_volume = 0
63              bin_frequencia = 0
64              for i in data['total_itens']:
65                  if x[i, b].solution_value() > 0:
66                      print(
67                          f"Item {i+1} volume: {data['volume'][i]}
68                          frequencia: {data['frequencia'][i]}")
69                      bin_volume += data['volume'][i]
70                      bin_frequencia += data['frequencia'][i]
71              print(f'Volume do Bin preenchido: {bin_volume}')
72              print(f'Maior Frequencia do Bin preenchido: {bin_frequencia}\n')
73              total_volume += bin_volume
74          print(f'Total de Volume Preenchido: {total_volume}')
75          print("\nAdvanced usage:")
76          print(f"Problem solved in {solver.wall_time():d} milliseconds")
77          print(f"Problem solved in {solver.iterations():d} iterations")
78
79      else:
80          print('O problema não possui uma solução ótima.')
81
82  if __name__ == '__main__':
83      main()
```

APÊNDICE B – Código Completo do Algoritmo para o Solucionador CP-SAT Escrito em Linguagem *Python*

```
"""Resolver o problema de multiplas mochilas usando o CP-SAT solver."""
from ortools.sat.python import cp_model

def main():
    data = {}
    data['volume'] = [
        432000, 800000, 588000, 425250, 784000, 3240000, 202500, 202500, 650000, 240000,
        560000, 432000, 800000, 588000, 425250, 784000, 3240000, 202500, 432000, 800000,
        588000, 425250, 784000, 3240000, 202500, 432000, 800000, 588000, 425250, 3240000]
    data['frequencia'] = [
        1.38, 5.76, 4.23, 4.89, 5.11, 2.19, 7.43, 7.18, 6.1, 1.68,
        4.84, 0.23, 2.09, 0.56, 1.14, 5.98, 2.19, 5.36, 2.55, 4.69,
        1.93, 1.45, 0.75, 3.37, 2.78, 1.49, 3.75, 1.43, 4.56, 2.91]

    assert len(data['volume']) == len(data['frequencia'])
    data['num_itens'] = len(data['volume'])
    data['total_itens'] = range(data['num_itens'])
    data['bin_capacidade'] = [5434000, 5434000, 5434000, 5434000, 5434000, 5434000,
                              5434000, 5434000, 5434000, 5434000, 5434000, 5434000]
    data['num_bins'] = len(data['bin_capacidade'])
    data['total_bins'] = range(data['num_bins'])

    # Criar o modelo para o Solver com o CP-SAT backend.
    model = cp_model.CpModel()

    # Variaveis.
    # x[i, b] = 1 se item i está no bin b.
    x = {}
    for i in data['total_itens']:
        for b in data['total_bins']:
            x[i, b] = model.NewBoolVar(f'x_{i}_{b}')

    # Restriçoes.
    # Cada item está alocado a no máximo um bin.
    for i in data['total_itens']:
        model.AddAtMostOne(x[i, b] for b in data['total_bins'])

    # A quantidade alocada em cada bin não pode exceder sua capacidade.
    for b in data['total_bins']:
        model.Add(
            sum(x[i, b] * data['volume'][i]
                for i in data['total_itens']) <= data['bin_capacidade'][b])

    # Objetivo.
    # Maximizar a frequencia total dos itens alocados.
    objective = []
    for i in data['total_itens']:
        for b in data['total_bins']:
            objective.append(
                cp_model.LinearExpr.Term(x[i, b], data['frequencia'][i]))
    model.Maximize(cp_model.LinearExpr.Sum(objective))

    solver = cp_model.CpSolver()
    status = solver.Solve(model)

    if status == cp_model.OPTIMAL:
```

```
57          print(f'Total de Frequencia Preenchido: {solver.ObjectiveValue()}')
58          total_volume = 0
59          for b in data['total_bins']:
60              print(f'Bin {b+1}')
61              bin_volume = 0
62              bin_frequencia = 0
63              for i in data['total_itens']:
64                  if solver.Value(x[i, b]) > 0:
65                      print(
66                          f"Item {i+1} volume: {data['volume'][i]}
67                          frequencia: {data['frequencia'][i]}"
68                      )
69                      bin_volume += data['volume'][i]
70                      bin_frequencia += data['frequencia'][i]
71              print(f'Volume do Bin preenchido: {bin_volume}')
72              print(f'Maior Frequencia do Bin preenchido: {bin_frequencia}\n')
73              total_volume += bin_volume
74          print(f'Total de Volume Preenchido: {total_volume}')
75
76          print("\nAdvanced usage:")
77          print(f"Problem solved in {solver.WallTime()*1000} milliseconds")
78
79      else:
80          print('O problema não possui uma solução ótima.')
81
82  if __name__ == '__main__':
83      main()
```