Maximizando o valor de carregamento: Uma abordagem com aspecto de otimizar geneticamente a alocação de produtos em espaços tridimensionais

Erick H. D. de Souza[1], **Lucas C. Dornelas**[1], **Marina B. Diniz**[1]

[1]Departamento de Computação - Centro Federal de Educação Tecnológica de Minas Gerais
Av. Amazonas 7675, Nova Gameleira, Belo Horizonte - MG - Brasil. CEP: 30510-000

erickhenriquedds@gmail.com, lucascdornelas@gmail.com, aniram.520@gmail.com

Abstract. *The Three dimensional knapsack is a variation of the Knapsack problem that aims to optimize the allocation of products in three-dimensional space. This work aims to present the Three dimensional knapsack aimed at maximizing profit and compare the results obtained through its application in the GUROBI solver and in the genetic algorithm implemented by the authors.*

Resumo. *O Problema do Contêiner é uma variação do Problema da Mochila que visa a otimização da alocação de produtos num espaço tridimensional. Esse trabalho tem por objetivo apresentar o Problema do Contêiner voltado a maximização do no valor alocado e comparar os resultados obtidos através de sua aplicação no solver GUROBI e no algoritmo genético implementado pelos autores.*

1. Introdução

O problema da alocação de produtos em espaços tridimensionais é um clássico dilema de otimização. Tal problema destina-se a colocar o maior número de objetos em um certo espaço, ou seja, determinar uma configuração física de objetos para serem dispostos em um determinado lugar visando maximizar o volume utilizado deste espaço. O problema pode ter outros critérios de análise, ou seja, objetivos, tais como a maximização do peso da carga e a maximização do lucro relativo à carga.

Uma das variações desse problema, é o *Bin-Packing Problem*, que é definido com a premissa onde toda a carga deve ser armazenada, utilizando um espaço finito de armazenamento, como por exemplo um contêiner, e possui como objetivo utilizar o menor número possível de contêineres para a realização do transporte de toda a carga. Tem-se também, o *Problema da Mochila*, que é reconhecido por possuir apenas uma única mochila (espaço de armazenamento) para efetuar a alocação dos itens. O objetivo é encher a mochila selecionando itens que proporcionem o melhor valor possível. Ademais, temos o *Problema das Multi-Mochilas*, que é uma variação do *Problema da Mochila*, onde agora temos à disposição mais de uma mochila para a alocação dos itens de acordo com algum determinado objetivo.

O presente trabalho irá abordar a variante *Problema do Contêiner* sobre o ponto de vista do *Problema das Multi-Mochilas*, utilizando contêineres de diferentes tamanhos e volumes. Será abordado uma técnica heurística de otimização para chegar a uma resolução aproximada do problema, a fim de chegar a uma solução viável. Para isso,

será utilizado um algoritmo genético, especializado para resolver o problema abordado nesse estudo, baseado nas ideias de (Chu and Beasley, 1998), para possuir maior controle sobre a diversidade da população. Instâncias foram criadas e resolvidas por esse algoritmo, cujos resultados individuais serão comparados com a respectiva solução ótima encontrada pelo solver (Gurobi Optimization, 2008) ao resolver o modelo matemático de programação linear inteira mista para o problema.

2. Trabalhos Relacionados

O *Problema do Contêiner* foi inicialmente estudado por (Gilmore and Gomory, 1965) e enunciado por (Dyckhoff, 1990), como um problema cujo objetivo é o armazenamento, de maneira eficiente, de objetos em meios de transporte, tais como um contêiner.

A solução desse problema visa atingir o carregamento ótimo, reduzindo o custo do transporte e aumentando a estabilidade e apoio da carga.

Segundo (Bortfeldt and Gehring, 2001) o *Problema do Contêiner* é diferenciado em vários tipos:

1. *Bin-Packing*;
2. *Knapsack*;
3. *Empacotamento de Pilhas*;
4. *Multi-Contêiner*;

E, assim como o *Problema da Mochila*, o *Problema do Contêiner* é um problema NP-difícil e, portanto, complexo de ser resolvido matematicamente e deterministicamente. Resolvê-lo computacionalmente também é complexo, por se tratar de um problema de otimização combinatória (Dantzig, 1957). Logo, fatores restritivos relacionados ao empilhamento de cargas, aos pesos, às dimensões ou aos valores monetários, dificultam a determinação da solução ótima.

3. Definição do Problema

Problemas de empacotamento têm aparecido constantemente nas indústrias, com aplicações principalmente nas cadeias de suprimento e logística, e sua principal forma de armazenamento é feita através de contêineres.

Os contêineres são muito usados na comercialização internacional por serem seguros, rápidos e, principalmente, baratos, o que minimaliza os custos de transporte. Podem ser transportados por meios rodoviários, ferroviários, aéreos ou marítimos, sendo o último o meio mais comum. São caracterizados como uma grande caixa de madeira, ferro ou aço, que só é aberto no momento do carregamento e posteriormente no local do destino. São fabricados de tamanhos diferentes, dependendo de seu tipo de carregamento e destinação. Este tipo de transporte é cobrado por unidade e não pela quantidade de produto transportado. O que apresenta vantagem de aproveitamento de volume. E, caso a alocação seja mal planejada, haverá prejuízos para o vendedor.

Considerando este cenário, suponhamos que uma determinada empresa, possua a sua disposição três contêineres para enviar produtos para uma determinada localização. Essa empresa possui como objetivo, carregar nestes contêineres, a maior quantidade possível dos seus produtos, sendo que a prioridade de alocação, são os produtos mais valiosos disponíveis em estoque.

As informações de volume e capacidade de peso suportada pelos contêineres (Marítimo), foram encaminhadas à empresa de acordo com a Tabela 1.

Tabela 1. Informações dos contêineres

Contêiner	Volume (m^3)	Capacidade de peso (kg)
1	28250	33.178945
2	28600	76.2864896
3	28750	67.6342784

No momento do carregamento, um funcionário da empresa, responsável por determinar quais produtos devem ser alocados em cada contêiner, recebe uma planilha contendo as informações dos itens disponíveis para envio, onde as informações mais relevantes destes produtos, são os dados de volume, preço, peso e tipo de cada produto. Os tipos dos produtos são dados pela sua categoria, podendo ser: alimento, veneno, produtos de limpeza, vestuário ou eletrodoméstico. Além desta planilha, o funcionário também deve-se atentar que alguns produtos não podem ser transportados com outros, como informa a Tabela 2, onde (1) demonstra que o produto pode ser transportado com outro produto e (0) caso contrário.

Tabela 2. Tabela de Tipos de Produtos

Tipo	Alimento	Veneno	Limpeza	Vestuário	Eletrodoméstico
Alimento	1	0	0	0	1
Veneno	0	1	0	0	1
Limpeza	0	0	1	0	1
Roupa	0	0	0	1	1
Eletrodoméstico	1	1	1	1	1

O despachante sabe que a soma dos pesos e dos volumes de todos os produtos alocados dentro de um determinado contêiner, não deve ultrapassar o peso e o volume do mesmo. Ele deve realizar o carregamento desses contêineres colocando a maior quantidade de produtos possíveis de modo a maximizar o valor dos produtos.

3.1. Modelo Matemático

Considerando as informações apresentadas e tendo como base o modelo proposto por (Chen et al., 1995), criou-se um modelo matemático escalável que encontra uma solução linear ótima. Neste modelo, considera-se que tanto o volume quanto o peso das caixas e dos contêineres sejam conhecidas. Eis as características do modelo:

Conjuntos:
I - conjunto de itens;
J - conjunto de mochilas;

Parâmetros:
v_i - volume do item $i \in I$
p_i - peso do item $i \in I$
m_i - valor do item $i \in I$

B_j - volume do contêiner $j \in J$
P_j - capacidade de peso do contêiner $j \in J$
$T_{ik} = \begin{cases} 1, & \text{se o produto } i \text{ puder compartilhar o contêiner com o produto } k \\ 0, & \text{caso contrário} \end{cases}$

Variáveis de decisão:
$X_{ij} = \begin{cases} 1, & \text{se o produto } i \text{ estiver no contêiner } j \\ 0, & \text{caso contrário,} \end{cases}$

$$\max_{X_{ij}} \sum_{i \in I} \sum_{j \in J} X_{ij} m_i \tag{1}$$

sujeito a:

$$\sum_{i \in I} X_{ij} v_i \leq B_j \; \forall \; j \in J \tag{2}$$

$$\sum_{i \in I} X_{ij} p_i \leq P_j \; \forall \; j \in J \tag{3}$$

$$\sum_{j \in J} X_{ij} \leq 1 \; \forall \; i \in I \tag{4}$$

$$\sum_{i \in I} \sum_{k \in I} X_{ij} + X_{kj} \leq 1, \; \forall \; j \in J : i \neq k, T_{ik} = 0 \tag{5}$$

O objetivo do modelo é expresso na função objetivo (1), ou seja, quer-se maximizar o valor dos produtos armazenados nos contêineres.

Contudo, o somatório tanto dos volumes (2) quanto dos pesos dos produtos (3) não deve ser maior que o volume e o peso do contêiner, respectivamente, para que seja possível realizar o carregamento sem que haja danos ao contêiner. Ademais, considerando uma lista onde não há repetições, ou seja, cada produto é único, não se pode carregar o mesmo produto em diferentes contêineres (4).

Por fim, tem-se a restrição quanto ao compartilhamento descrito na seção 3, onde um determinado tipo de produto não pode ser transportado no mesmo contêiner que outro determinado tipo de produto (5).

4. Metodologia

4.1. Algoritmo Genético

Para encontrar uma possível solução para o problema definido anteriormente na seção 3, utilizou-se o algoritmo genético. Esse algoritmo emprega técnicas de busca e otimização, sendo capaz de lidar com problemas combinatórios, como os problemas de alocação de produtos. A ideia é criar uma população de indivíduos que vão se reproduzir e competir pela sobrevivência, onde os melhores irão sobreviver e transmitir suas características à sua prole, até que se chegue a uma solução próxima do resultado ótimo (Holland, 1975), ou

seja, até encontrar a alocação ideal dos produtos em diferentes contêineres, maximizando o valor dos produtos armazenados.

O algoritmo foi implementado em Python 3.11.2, utilizando-se o sistema operacional Ubuntu 23.04, processador Intel® Core™ i7-10700K × 16 e 32GB de memória RAM.

Nas subseções abaixo, explica-se os passos seguidos para implementar o algoritmo, baseado no modelo matemático descrito na seção 3.1.

4.1.1. Representação Cromossomial

Inicialmente, define-se uma representação cromossomial adequada para o problema. Neste caso, cada cromossomo será uma sequência de genes que representam a alocação dos produtos em diferentes posições dos contêineres. De modo que, um gene na posição n, pode assumir um valor m inteiro que ira definir em qual contêiner o produto está, caso for 0 ele não será adicionado em nenhum contêiner.

4.1.2. Função de Avaliação

Uma função de avaliação (ou função de fitness) é necessária para medir o quão bom é um indivíduo (solução). No contexto deste problema, a função de avaliação deve considerar a utilização eficiente do espaço nos contêineres, a capacidade de peso do contêiner e as restrições de compartilhamento dos tipos produtos.

Portanto, a função de avaliação atribuirá uma pontuação aos indivíduos com base nesses critérios, caso a solução não respeite as restrições irá receber o valor zero.

4.1.3. Operadores Genéticos

Os operadores genéticos são utilizados para criar novas soluções através da manipulação dos cromossomos. No algoritmo proposto, utiliza-se os seguintes operadores:

- **Seleção**: Seleciona indivíduos mais aptos para a reprodução com base em sua pontuação de fitness.
- **Cruzamento**: Realiza o cruzamento entre dois indivíduos selecionados, combinando partes de seus cromossomos para criar novos indivíduos.
- **Mutação**: Introduz pequenas alterações aleatórias nos cromossomos dos indivíduos, a fim de explorar novas regiões do espaço de busca.

4.2. Processo de Otimização

O processo de otimização consiste na aplicação iterativa dos operadores genéticos para evoluir uma população de indivíduos ao longo das gerações. Inicialmente, uma população inicial de soluções é gerada aleatoriamente. Em seguida, os operadores genéticos são aplicados, permitindo que os indivíduos mais aptos sobrevivam e se reproduzam.

O algoritmo continua a evoluir a população até que um critério de parada seja atingido, como um número máximo de gerações ou a convergência para uma solução satisfatória.

Após a execução do algoritmo genético, os melhores indivíduos obtidos ao longo das gerações representam soluções aproximadas para o problema de alocação de produtos em contêineres. Os resultados podem ser avaliados em termos de utilização do espaço e quantidade.

4.3. Pseudo código

Abaixo está o pseudo código do algoritmo implementado pelos autores.

Algorithm 1 Algoritmo Genético

procedure ALGORITMO GENÉTICO
    Inicialização:
    $populacao \leftarrow$ gerarPopulacao(tamanhoPopulacao)
    $melhorIndividuo \leftarrow$ encontrarMelhorIndivíduo(population)
    for $i \leftarrow 1$ to numGerações **do**
        Seleção:
        $pais \leftarrow$ selecionarPais(populacao)
        Recombinação:
        $filhos \leftarrow$ recombinar(pais)
        Mutação:
        $filhos \leftarrow$ mutar(filhos)
        Avaliação:
        $populacao \leftarrow$ filhos
        Atualização:
        $melhorIndividuo \leftarrow$ encontrarMelhorIndivíduo(populacao)
    end for
    Retorne: $melhorIndividuo$
end procedure

5. Resultados

Para obter uma base mínima de dados para comparação, executa-se algumas instâncias variando o número de produtos disponíveis a uma determinada faixa fixa e imutável de contêineres, que pode ser visualizada na Tabela 3 e na Tabela 4. Para uma análise mais profunda, analisa-se algumas destas instâncias separadamente.

Tabela 3. Análise de resultados do solver

Quantidade de produtos	Tempo de execução(s)	Valor ótimo(R$)
500	41.9	4295.50
1000	54.3	5442.23
2000	265.3	7048.87
3000	840.7	7675.69
4000	1192.9	8729.38
5000	1921.2	9077.62
6000	3526.5	9729.75
7000	5497.7	9978.24

Tabela 4. Análise de resultados do algoritmo genético

Quantidade de produtos	Tempo de execução(s)	Valor encontrado(R$)	GAP(%)
500	3.2	3966.70	7.65
1000	18.2	4705.41	13.53
2000	47.1	6374.98	9.56
3000	49.3	6122.43	20.20
4000	132.0	6813.20	21.94
5000	250.3	7029.08	22.56
6000	369.5	7375.18	24.19
7000	532.6	8388.74	15.92
8000	1001.2	9896.36	0.00

5.1. 500 produtos

Nesta primeira análise, separa-se 500 produtos disponíveis para alocação.

Como pode ser visto na Tabela 5, o solver encontrou o valor ótimo correspondente a R$ 4295.50 em 41.9s. Enquanto que o algoritmo genético encontra uma solução viável de R$ 2859.78 em um tempo menor, 3.2s.

Tabela 5. Análise de resultados para 500 produtos

Algoritmo genético		Solver Gurobi	
Tempo de execução(s)	Valor encontrado (R$)	Tempo de execução(s)	Valor ótimo(R$)
3.2	2859.78	41.9	4295.50

A Figura 1 demonstra a variação dos indivíduos a cada geração do algoritmo genético.

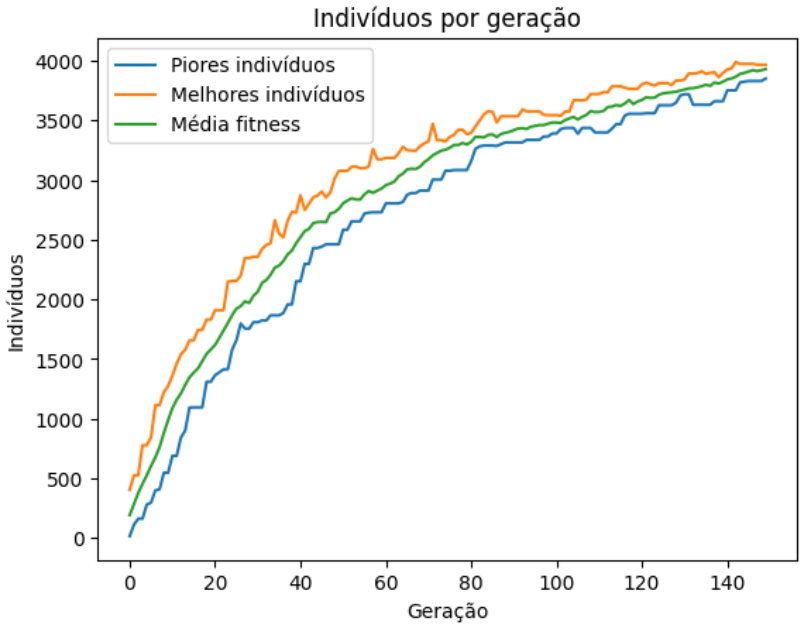

Figura 1. Análise dos indivíduos em 500 produtos

Apesar de ser bem mais eficiente (encontrando uma solução em menos tempo), o algoritmo não encontrou o mesmo resultado que o solver. Isso ocorre devido a premissa de que o algoritmo genético é mais rápido do que solver, mas sem garantia de otimalidade.

5.2. 3000 produtos

Para 3000 produtos, o solver levou 840.7s para chegar na solução ótima, enquanto o algoritmo genético chegou a uma solução próxima a ótima em um tempo 94.14% menor. Isso pode ser observado na Tabela 6.

Tabela 6. Análise de resultados para 3000 produtos

Algoritmo genético		Solver Gurobi	
Tempo de execução(s)	Valor encontrado (R$)	Tempo de execução(s)	Valor ótimo(R$)
49.3	6122.43	840.7	7675.69

A Figura 2 demonstra a variação dos indivíduos a cada geração do algoritmo genético:

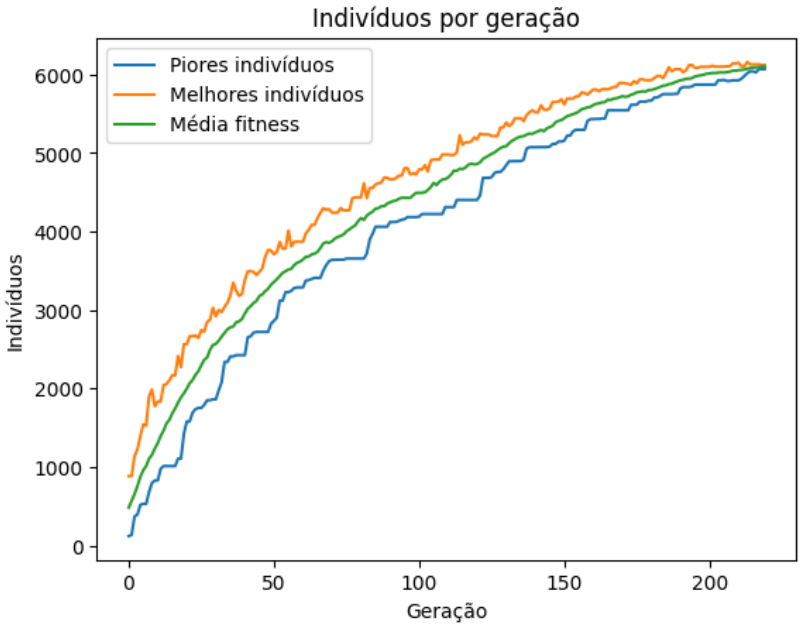

Figura 2. Análise dos indivíduos em 3000 produtos

5.3. 7000 produtos

Neste caso, com 7000 produtos, observa-se que a variação dos indivíduos e o tempo de execução está tendendo a se aproximar ao se aumentar o número de gerações, ao comparar com os resultados anteriores, o que pode ser observado na Tabela 7.

Tabela 7. Análise de resultados para 7000 produtos

Algoritmo genético		Solver Gurobi	
Tempo de execução(s)	Valor encontrado (R$)	Tempo de execução(s)	Valor ótimo(R$)
532.6	8388.74	5497.7	9978.24

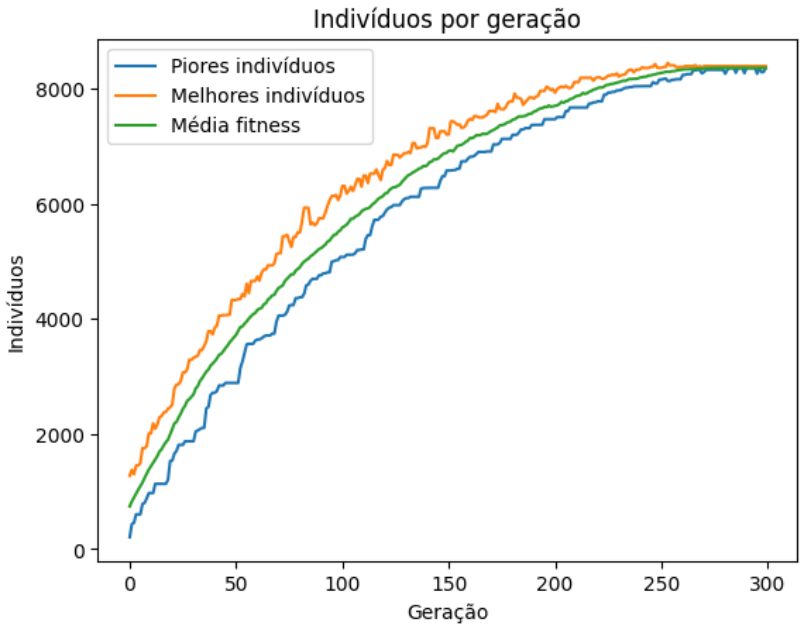

Figura 3. Análise dos indivíduos em 7000 produtos

5.4. 8000 produtos

Para 8000 produtos, não foi possível obter resultados satisfatórios com o solver, visto que a quantidade de memória RAM necessária para sua execução não era compatível com a máquina utilizada para a extração dos dados.

Sendo assim, observamos que a partir de uma determinada quantidade de produtos, se torna inviável a resolução do mesmo pelo solver. O que delimita o ponto de uso do algoritmo genético, que encontrou uma solução viável com 1001s.

Por fim, observa-se que os resultados tendem a manter o mesmo padrão independentemente do número de produtos aplicados. Podendo validar essa afirmativa com base nos dados de todas as amostras obtidas, vide Tabelas 3 e Tabelas 4.

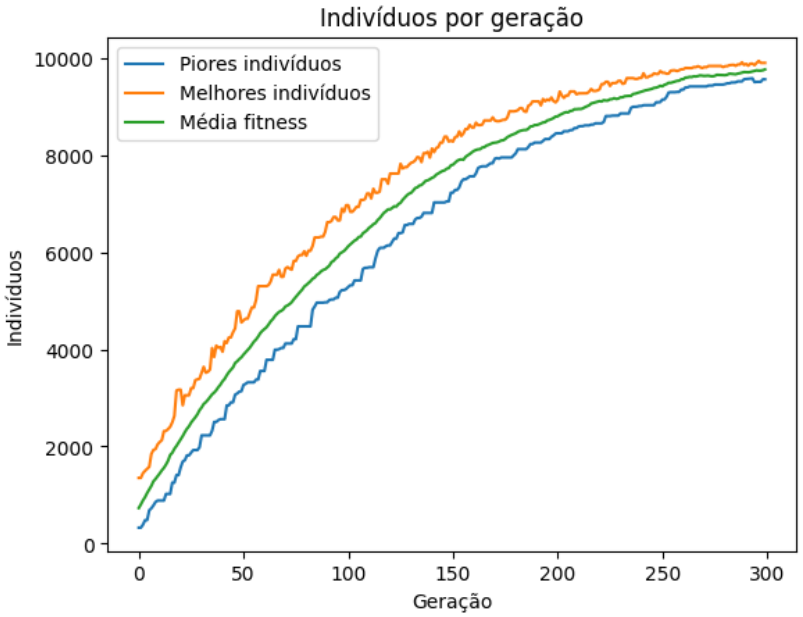

Figura 4. Análise dos indivíduos em 8000 produtos

6. Considerações Finais

Após uma minuciosa análise das instâncias, torna-se evidente que o solver é capaz de entregar uma solução ótima dentro de um prazo razoável quando lidamos com um número reduzido de itens. No entanto, à medida que o volume de produtos aumenta, o solver ainda é capaz de encontrar a melhor solução, embora em um período de tempo significativamente maior, e necessitando de um grande poder de processamento dos dados. Assim, tornando-se inviável quando atinge o valor máximo, no caso do modelo estudado e na máquina utilizada, 8000 itens. Além disso, considerando a natureza do *Problema da Mochila*, quanto maior o número de restrições, maior a complexidade do problema, ou seja, maior o tempo de execução. Isso pode ser observado, por exemplo, no balanceamento da carga interna do contêiner ou alocação de produtos com base na ordem de entrega final.

Por outro lado, o algoritmo genético apresenta um tempo de execução menor em comparação ao solver, porém não entrega uma solução ótima, e sim uma viável. Sendo assim, quando o problema envolve a análise de uma grande quantidade de produtos, a opção de se utilizar o algoritmo genético é mais factível ao se comparar com o método via solver.

Dado que o prazo é um fator restritivo, o uso do solver se torna inviável para certos propósitos. Isso pode ser observado, por exemplo, na determinação de alocação de produtos em um veículo de transporte para a realização de entregas de produtos adquiridos pela internet. Segundo (Estado de Minas, 2022), no ano de 2021, 353 milhões de entregas foram realizadas em todo o território nacional, ou seja, a cada dia do ano, aproximadamente 980.555 entregas ocorreram, apenas de itens adquiridos via e-commerce. Sendo assim, empresas do ramo buscam encontrar as melhores soluções possíveis que supram a necessidade do vendedor, colocando em rota todos os produtos a serem entregues, e a demanda do consumidor, recebendo o seu produto no prazo estabelecido, e se possível, com antecedência, para aumentar a satisfação do cliente perante o comerciante.

Esses fatores desempenham um papel crucial na tomada de decisão das empresas por optar pelo uso de heurísticas para alocar suas entregas. Uma vez que, embora não garanta a solução ideal, elas oferecem uma alternativa aceitável que pode ser alcançada em um tempo mais curto.

Referências

A. Bortfeldt and H. Gehring. A hybrid genetic algorithm for the container loading problem. *European Journal of Operational Research*, 131:143–161, 2001.

C. S. Chen, S. M. Lee, and Q. S. Shen. An analytical model for the container loading problem. *European Journal of Operational Research*, 80:68–76, 1995.

P. C. Chu and J. E. Beasley. A genetic algorithm for the multidimensional knapsack problem. *Journal of heuristics*, 4:63–86, 1998.

G. Dantzig. Discrete-variable extremum problems. *Operations Research*, 5:266–277, 1957.

H. Dyckhoff. A typology of cutting and packing problems. *European Journal of Operational Research*, 44:145–159, 1990.

Estado de Minas. Com pandemia, vendas pela internet crescem 272021, 2022. URL `https://www.em.com.br/app/noticia/economia/2022/02/02/internas_economia,1342064/com-pandemia-vendas-pela-internet-crescem-27-e-atingem-r-161-bi-em-2021.shtml`.

P. C. Gilmore and R. E. Gomory. Multistage cutting problems of two and more dimensions. *Operations Research*, 13:94–119, 1965.

Gurobi Optimization, 2008. URL `https://www.gurobi.com/`.

J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, MI, 1975.

G. Marítimo. URL `https://www.guiamaritimo.com.br/utilidades/tipos-containers`.