Modificações na heurística George & Robinson para carregamento 3D

Eduardo Villa Koga, Alexandre Checoli Choueiri

***Abstract*—Neste artigo, foi abordado o Problema de Carregamento de Contêineres *(Container Loading Problem - CLP)* utilizando, para sua resolução, a heurística de George e Robinson (G&R) e duas modificações aplicadas nela. Foi feita uma explicação da heurística de G&R sob nova ótica, utilizando fluxogramas e imagens construídas a partir do artigo original. Duas modificações foram então propostas na etapa de seleção do espaço trabalhado a cada iteração, gerando assim dois novos métodos. Estes, juntamente com a heurística original, foram implementados computacionalmente, testados em uma instância que fornece um amplo conjunto de problemas e seus resultados foram comparados. Enquanto uma das modificações não apresentou vantagens frente ao método de G&R, a outra com frequência encontrou melhores soluções, ampliando o volume ocupado dentro do contêiner.**

***Index Terms*—container, loading, optimization, meta-heuristics.**

I. Introdução

O Movimento de um produto, material ou suprimento que faz com que ele chegue de um local até o outro é uma atividade essencial para qualquer empresa, mesmo as que oferecem serviços intangíveis. Essa movimentação de produtos é uma atividade da logística de suprimentos, responsável pela determinação de quantidades, volumes, datas e rotas envolvidos na operação. Como são decisões complexas e usualmente intricadas, um bom gerenciamento das atividades logísticas pode acarretar em ganhos para a empresa, bem como em operações mais confiáveis [1]. A subárea da logística responsável pela maior parcela dos custos industriais se da pelas decisões de transporte.

Os transportes, no contexto da logística, exercem um importante papel nas economias de hoje e se referem a efetivamente levar o material ao seu destino de modo aéreo, terrestre ou aquático. Em diversos setores, os recursos que uma empresa destina aos transportes representam grande parte de seu custo logístico total e geram alto impacto no nível de serviço ao cliente. Dessa maneira, analisar e planejar os transportes e seus custos é fundamental para o gerenciamento do sistema logístico como um todo. Dentre as maneiras de se otimizar os transportes e reduzir seus custos temos: a definição de rotas para veículos, a consolidação e despacho de cargas e o carregamento de contêineres [2].

Para realizar cada uma dessas otimizações, recorremos a diferentes métodos, a depender da dificuldade e especificidade de cada problema. O Problema de Carregamento de Contêineres *(Container Loading Problem - CLP)* é caracterizado como um problema NP - difícil (de alta complexidade) e os métodos mais tradicionais para resolvê-lo são os algoritmos exatos, as heurísticas e as meta-heurísticas. No entanto, algoritmos exatos já não capazes de encontrar uma solução para o CLP em um tempo computacional aceitável. Por esse motivo, lança-se mão de algoritmos heurísticos, que não garantem uma solução ótima, porém conseguem soluções de boa qualidade em tempos aceitáveis [3].

Uma das heurísticas pioneiras para tratar do CLP é a proposta no artigo de George e Robinson (G&R), em 1980 [4]. O método apresentado nesse trabalho [5] se caracteriza pelas ideias de dividir o contêiner em varias camadas ao longo do seu comprimento, formar paredes horizontais com caixas do mesmo tipo, utilizar critérios de seleção para escolher a caixa que será colocada inicialmente em cada camada e em cada espaço, considerar novos espaços gerados quando uma camada não é totalmente preenchida e juntar esses espaços para otimizar o arranjo final. Essa é uma heurística que, comparada a outras da literatura, possui simplicidade, flexibilidade e facilidade de implantação [6].

Nesse trabalho, realizamos uma proposta de melhoria na heurística de G&R, adicionando uma componente estocástica em seu processo de seleção de espaços, o que possibilita a geração de soluções distintas a cada nova reinicialização do método. Dessa forma, a heurística proposta se enquadra na família de métodos *multi-start*. Tanto esta quanto o método original foram implementados computacionalmente utilizando o VBA (*Visual Basic for Applications*), linguagem de programação acessível para todos os usuários que possuem o software Microsoft Excel no computador.

Este artigo está organizado da seguinte forma: na Seção II são apresentados trabalhos correlatos para o CLP. Na Seção III a heurística de G&R é explicada, bem como as modificações realizadas na mesma. Finalmente, na Seção IV apresentamos os resultados computacionais da nova proposta, e concluímos o trabalho na Seção V.

II. Revisão de literatura

No Problema de Carregamento de Contêineres usual, tem-se inicialmente um conjunto de itens tridimensionais e busca-se inseri-los dentro de um contêiner retangular tridimensional de modo a maximizar uma função objetivo, que comumente é o volume ou o lucro total associado aos itens carregados. As restrições comuns a praticamente todas as variantes do CLP são:

1) dois itens não podem sobrepor um ao outro (ou seja, ocupar o mesmo espaço)

This paper was produced by the IEEE Publication Technology Group. They are in Piscataway, NJ.

Manuscript received November 1, 2024; revised November 1, 2024.

2) todos os itens carregados devem estar totalmente dentro do contêiner (evitando situações em que parte de um item está fora) [7]

Muitas variações do problema surgem a partir disso, adicionando diversas restrições ou alterando a função objetivo.

Os métodos exatos propostos para resolver o CLP são escassos se comparados aos heurísticos, por exigirem um grande tempo computacional. Ainda assim, podemos encontrar alguns métodos relevantes na literatura. Chen et al. [8] propuseram um modelo que considera as rotações de itens e resolve uma versão do problema em que não necessariamente todos os contêineres tenham as mesmas dimensões, discutindo como adaptar o modelo para outras restrições. Por outro lado, Hifi et al. [9] não incluíram as rotações em seu modelo ao mesmo tempo em que consideraram que todos os contêineres eram idênticos e que a função objetivo buscava minimizar o número total deles. Tsai et al. [10] basearam sua formulação na de Chen et al. [8], mas agora considerando apenas um contêiner sem dimensões fixas e buscando minimizar o volume total deste após todas as caixas serem carregadas. Já Paquay et al. [11] fizeram a formulação do problema considerando várias restrições especificas para adaptá-lo ao transporte aéreo e utilizando coordenadas próprias, diferente da maioria dos modelos, para indicar as posições de caixas.

Em [12], os autores consideram restrições de rotação de caixas, um sistema de coordenadas no centro da caixa ao invés dos cantos, e se utilizaram de um método heurístico em parte da resolução, que tinha por objetivo maximizar o volume das caixas carregadas em um único contêiner. Considerando o mesmo objetivo, Junqueira et al. [13] desenvolveram um método em que, para cada caixa, são calculados todos os múltiplos de suas dimensões, e então, apenas são alocadas caixas, sem considerar rotacionamentos, em coordenadas que correspondam a algum desses múltiplos. Já Martello et al. [14] propuseram um método exato diferente dos demais em que é utilizado o *Branch and Bound* para maximizar o volume carregado em um único contêiner. Em resumo, o método ordena as caixas do maior para o menor volume e gera a árvore do *Branch and Bound* a partir da combinação das caixas com todas as posições possíveis, começando do fundo do contêiner e indo até sua parte frontal.

Diversas meta-heurísticas conhecidas também foram utilizadas para resolver o CLP. Araya et al. [15] e Wang et al. [16] utilizaram algoritmos baseados no *beam-search* (uma otimização do *branch-and-bound* em que apenas o melhor ramo é expandido). Já nos trabalhos de Ramos et al. [17], Bortfeldt e Gehring [18], Jamrus et al. [19] e Zheng et al. [20] são utilizados *Genetic Algorithms* (Algoritmos Genéticos), enquanto Bayraktar et al. [21] trabalharam com *Artificial Bee Colony* no algoritmo proposto. *Simulated annealing* por sua vez está presente nos trabalhos de Dereli e Sena Das [22], Mostaghimi et al. [23], Sheng et al. [24] e Mack et al. [25]. Algoritmos que utilizam *tabu search* foram propostos por Bortfeldt et al. [26], Gendreau et al. [27], Tarantilis et al. [28] e Liu et al. [29].

Já as heurísticas mais conhecidas para encontrar uma solução para o CLP trabalham com diferentes métodos de arranjo de caixas no carregamento, sendo uns dos mais relevantes a formação de paredes, formação de camadas ou a junção dessas duas. A heurística que trouxe a ideia de formar paredes foi a proposta por G&R [5] e já resumida anteriormente. A partir da estrutura base dela, Bischoff e Marriott [30] desenvolveram um método em que são realizadas mudanças nas etapas de escolha de itens e ordem de carregamento, gerando assim 14 novas heurísticas, e então todas elas são aplicadas a um mesmo problema e o melhor resultado é escolhido. Seguindo com a formação de paredes, Moura e Oliveira [31] aplicaram duas mudanças na heurística de G&R que restringem as dimensões de largura de novos espaços criados a cada iteração, assim aumentando a estabilidade do carregamento.

Muitas vezes, uma heurística é utilizada para gerar uma solução inicial para o problema, e a partir dela, uma meta-heurística é aplicada para otimizar essa solução. É o caso do trabalho de Gehring e Bortfeldt [32] em que é utilizada uma abordagem de construção de pilhas. Os itens são dispostos um cima do outro formando pilhas em que a base de cada um está totalmente sustentada pela superfície do item abaixo dele, e então um algoritmo genético é utilizado para realizar o arranjo das pilhas no contêiner.

Outra abordagem heurística para o problema é a de arranjo por blocos, em que itens se juntam formando blocos e então estes são inseridos no contêiner. Bortfeldt et al. [26] e Mack et al. [25] aplicaram essa abordagem em seus respectivos trabalhos, assim como Eley [33], que propôs uma metodologia em que os blocos são formados por itens idênticos e então um algoritmo guloso é utilizado para juntá-los, gerando assim uma solução que é otimizada por uma árvore de busca. Já no trabalho de Fanslau [34] é aplicada uma melhoria de diminuição no espaço interno vago dos blocos, enquanto nos de Zhu e Lim [35] e Zhang et al. [36] foram produzidos resultados interessantes utilizando simultaneamente blocos formados apenas por itens idênticos e blocos formados por itens diferentes.

Por sua vez, Morabito e Arenales [37] trouxeram um método diferente, em que o contêiner é dividido em várias partes como se tivesse sido cortado por uma guilhotina e cada uma dessas só deve receber um item. Em seguida, todas as partes são dispostas em um diagrama para avaliar as possibilidades de carregamento, escolhendo então o melhor resultado .

Diferentemente das heurísticas anteriores, os trabalhos de Ngoi et al. [38] e Huang e He [39] não se utilizam de arranjos ou de ordem de carregamento (do fundo para a frente, ou do piso para o teto). No primeiro, é utilizada uma matriz de três dimensões para armazenar cada caixa carregada no contêiner e cada espaço vago, então o algoritmo seleciona um item e um espaço e o preenche iterativamente. Já no segundo trabalho, o algoritmo carrega o contêiner priorizando caixas que preencham cada espaço da maneira mais compacta possível, priorizando os cantos e os pequenos espaços internos que surgem entre itens durante o carregamento.

III. DESENVOLVIMENTO

Nesta seção, explicaremos a heurística de G&R e a modificação feita na mesma. Para isso, primeiramente de-

screvemos o método de formação, separação e seleção de espaços no algoritmo.

A. A separação do espaço

Na heurística de G&R, um espaço é uma fração da área interna do contêiner, com suas faces paralelas às paredes deste e sempre em formato de bloco retangular. Para facilitar o entendimento, iremos arbitrariamente considerar que todo espaço é representado por uma sêxtupla: L (*length* ou comprimento), W (*width* ou largura), H (*height* ou altura), x, y e z. As três primeiras são as variáveis de dimensão, indicam o tamanho das dimensões do espaço, e as três últimas são as variáveis de posição, indicam a posição do ponto no canto inferior esquerdo do fundo do espaço em relação ao ponto no canto inferior esquerdo do fundo do contêiner, como em um sistema de coordenadas em que esse último é o ponto (0,0,0). A Figura 1 ilustra o espaço inicial utilizado na heurística e sua representação, note que para esse espaço x, y e z valeriam 0 e L, W e H teriam os valores das dimensões do contêiner, ou seja, a representação do espaço fica: $(x, y, z, L, W, H) = (0, 0, 0, L, W, H)$.

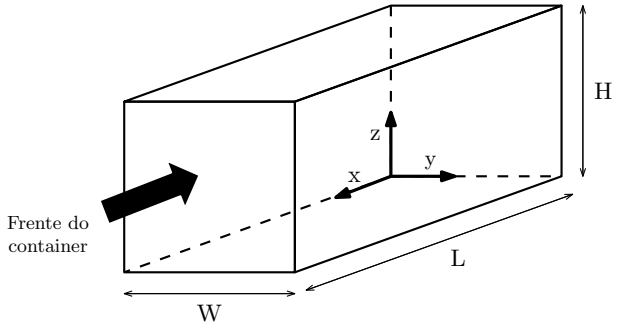

Fig. 1: Representação do espaço inicial

Na heurística de G&R, temos uma lista em que cada item é um espaço que será preenchido, inicialmente essa lista contém somente um item, que é o espaço inicial citado, e, conforme as iterações acontecem, novos espaços são formados e adicionados na lista, enquanto outros são removidos dela. A cada nova iteração, o último espaço da lista é selecionado e nele são carregadas as caixas, partindo do seu canto inferior esquerdo do fundo. Com a parede de caixas formada, o espaço selecionado pode ser dividido em até 3 novos espaços, que nomearemos como Espaço L (comprimento), Espaço W (largura) e Espaço H (altura), correspondendo ao posicionamento de cada um em relação às caixas carregadas. A Figura 2 ilustra esse procedimento. Após essa divisão, os três novos espaços são adicionados na lista na seguinte ordem: primeiro o Espaço L, depois o W e por fim o H e, após isso, o espaço utilizado na iteração é retirado da lista.

B. A heurística de G&R

A heurística de G&R será explicada seguindo dois fluxogramas que a ilustram, construídos a partir do artigo original. O fluxograma A (Figura 3) mostra a seleção dos espaços, a junção dos mesmos e a escolha do tipo de caixa, enquanto o

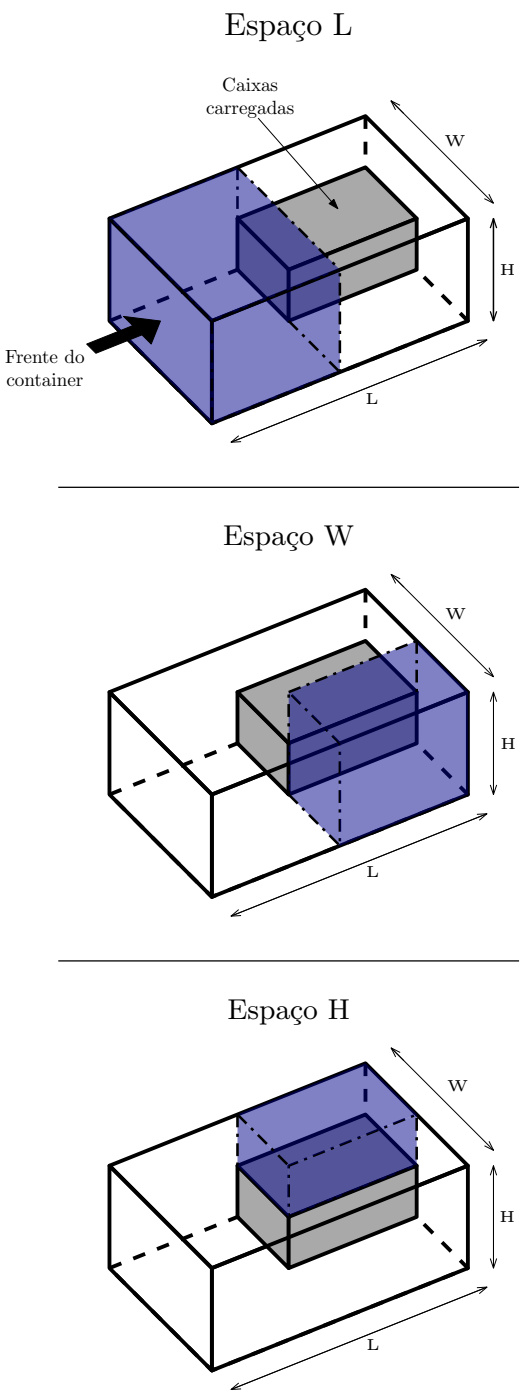

Fig. 2: Formação dos 3 espaços

fluxograma B (Figura 4) trata da rotação de caixas e criação de novos espaços.

O método se baseia em realizar o carregamento por camadas. Uma camada é uma seção do comprimento do contêiner, e pode ser entendido como um espaço $(x, y, z, L, W, H) = (x, 0, 0, L, Wco, Hco)$, sendo Wco e Hco os valores da largura e altura do contêiner, respectivamente. O comprimento (L) de uma camada é determinado pelo comprimento da primeira caixa colocada nele, e uma nova camada só será formada quando a última for preenchida até não restarem mais opções de caixas para serem carregadas

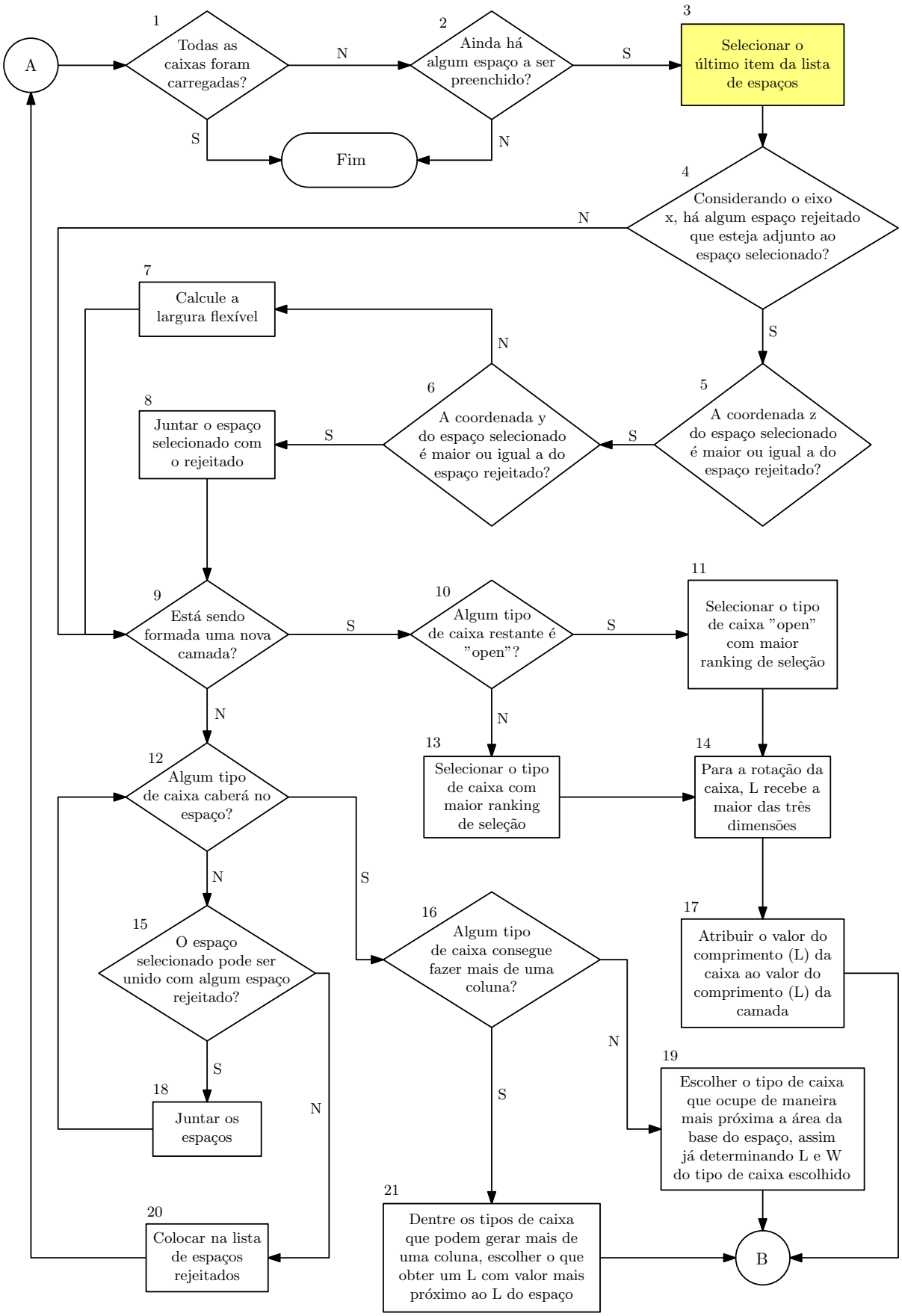

Fig. 3: Fluxograma A

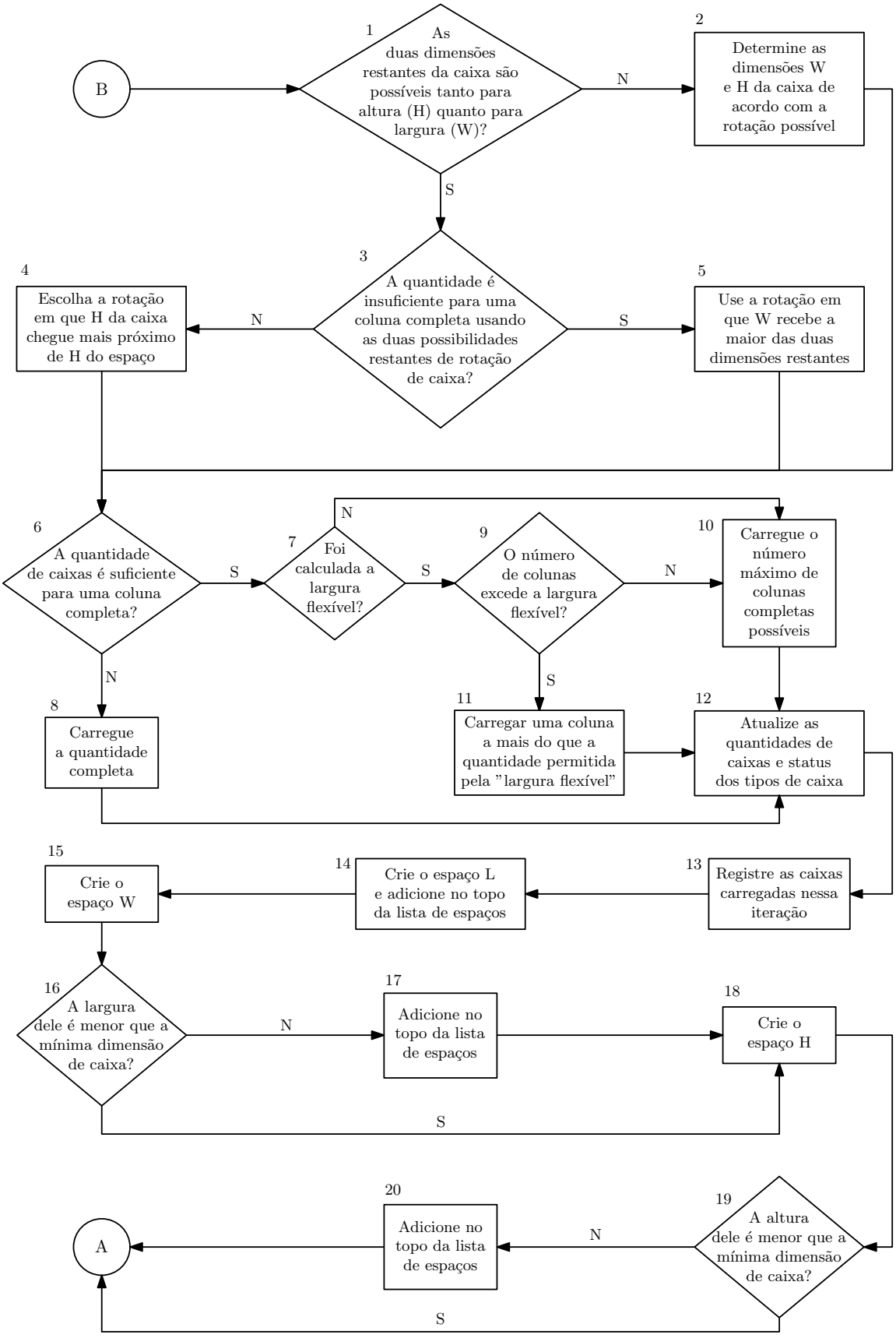

Fig. 4: Fluxograma B

nela. Esse procedimento é ilustrado na Figura 5.

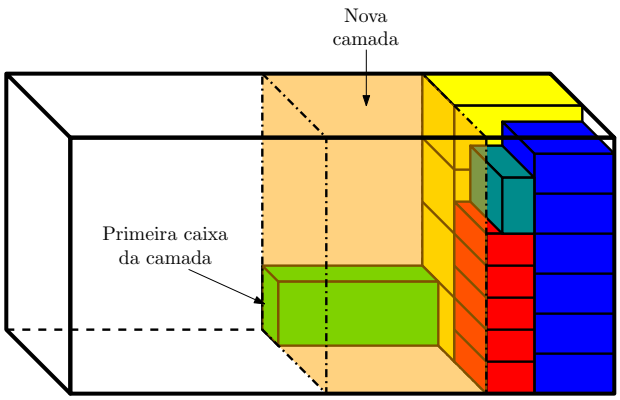

Fig. 5: Formação de uma camada

O início da heurística ocorre em A, no fluxograma A, as decisões em 1 e 2 ilustram os critérios de parada: caso todas as caixas já tenham sido carregadas ou caso não haja mais nenhum espaço que possa ser preenchido (não ter mais nenhum item na lista de espaços). Não satisfeitos os critérios de parada, temos uma nova iteração. O primeiro passo (3) é selecionar o último item da lista de espaços. Antes de preenchê-lo, verificamos se pode ser feita a junção deste com algum espaço rejeitado.

No decorrer do algoritmo, ocorrem situações em que tentamos preencher um espaço que tenha um comprimento (L) muito curto, e não encontramos uma caixa disponível que caiba no mesmo. Este espaço deve ser então retirado da lista, porém ao invés de descartá-lo, o adicionamos em uma lista de espaços rejeitados. Na continuação do método, pode chegar um momento em que o espaço selecionado na iteração pode ser unido com um desses espaços rejeitados, assim ampliando suas dimensões e aproveitando lacunas deixadas durante o carregamento, como ilustrado na Figura 6.

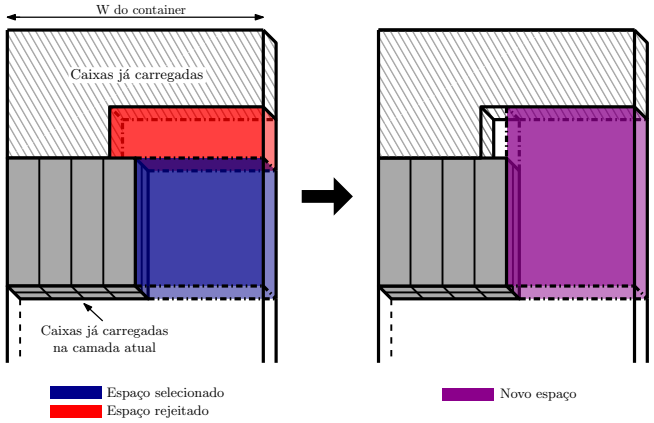

Fig. 6: Junção de espaços

Para essa junção ocorrer, primeiramente dois critérios precisam ser atendidos: os espaços serem adjuntos (terem uma face em comum no sentido do comprimento) e a variável z do espaço selecionado ser maior ou igual a do espaço rejeitado, indicados na decisões 4 e 5. Com isso, temos dois cenários possíveis: se a variável y do espaço selecionado for maior ou igual a do espaço rejeitado, realizamos a junção, mas se for menor, calculamos uma ”largura flexível”, que é obtida pela diferença entre y do espaço rejeitado e a soma de y e W do espaço selecionado (que gera a posição em y da sua face ”da direita”). Após isso, guardamos esse valor, que será útil no carregamento das caixas.

Cada caixa possui três dimensões, referentes às medidas das suas arestas e que são fornecidas nos dados do problema. Agrupamos caixas com as mesmas dimensões em um mesmo grupo, e a isso chamamos de tipo de caixa. A partir disso, utilizamos na resolução do CLP os dados das quantidades de caixas pertencentes a cada tipo. Tal como os espaços, as caixas também são representadas pelas variáveis de posição x, y e z e pelas variáveis de dimensão L, W e H, mas agora adicionaremos uma variável de quantidade, que indica quantas caixas daquele tipo ainda não foram carregadas, e uma variável de status, que indica se aquele tipo é ou não *”open”*.

No início, todos os tipos de caixa são *”não-open”*, e quando um deles é selecionado e algumas de suas caixas são carregadas, ele passa a ser *”open”*. Essa variável indica, então, se aquele tipo já foi utilizado ou não. Além disso, cada uma das dimensões de uma caixa (referentes as três medidas de suas arestas) é atribuída a uma de suas variáveis de dimensão (L, W ou H), assim determinando sua rotação. Note que em duas iterações distintas, um mesmo tipo de caixa pode assumir diferentes valores de L, W e H, indicando que de uma para outra foram utilizadas rotações diferentes. A Figura 7 ilustra um exemplo de uma caixa cujas dimensões (medidas das arestas) são 3, 6 e 9. Observe as possíveis 6 rotações que ela pode assumir, em cada uma, as variáveis de dimensão (L, W, H) recebem uma diferente combinação de valores.

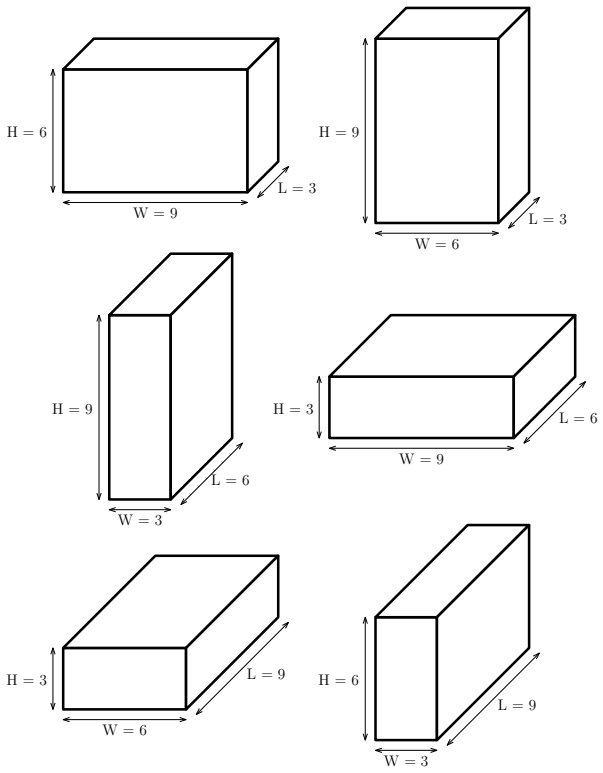

Fig. 7: Possíveis rotações de uma caixa

A partir da decisão em 9 no fluxograma A até o final deste, temos a escolha do tipo de caixa que será carregada na iteração. Caso esteja sendo formada uma nova camada, temos um ranking de seleção com três critérios: O primeiro se refere a menor dimensão de cada tipo de caixa, aquele que tiver a maior delas será escolhida. Caso neste critério ocorra um empate entre dois ou mais tipos, escolhemos entre eles o que tiver a maior quantidade de caixas restantes e, caso novamente obtivermos um empate, verificamos a maior dimensão de cada tipo empatado e escolhemos aquele que tiver a maior delas. Priorizamos tipos de caixa *open* para iniciar uma nova camada, aplicando o ranking de seleção nestes para a escolha. Se não houverem mais tipos de caixa *open* disponíveis, então o ranking é aplicado entre todos os tipos que ainda tiverem uma quantidade remanescente.

Quando o espaço não for o de formação de uma nova camada, as etapas a partir da decisão em 12 no fluxograma A mostram qual tipo de caixa escolher nesses casos. A variável L já é determinada logo após ser escolhido o tipo de caixa, como podemos ver em 14, 19 (caso em que W também já é determinada) e 21.

Na sequência, passamos para o fluxograma B, onde as etapas de 1 até 5 mostram a escolha da rotação que será utilizada. Com a variável L já determinada, temos apenas dois valores de dimensão restantes e, consequentemente, duas rotações diferentes podem ser formadas. No caso em que as duas são possíveis para o espaço selecionado, os itens 3, 4 e 5 definem os critérios para fazer a escolha entre as duas.

Do item 6 ao 13 no fluxograma B, é tratado o carregamento das caixas do tipo escolhido e na rotação determinada. As caixas são sempre alocadas uma em cima da outra, formando uma ou mais colunas, e estas são dispostas uma ao lado da outra (no sentido do eixo y) a partir do canto esquerdo inferior do fundo do espaço, formando assim uma parede. Na Figura 6, podemos observar esse procedimento nas ”Caixas já carregadas na camada atual”. Voltando ao fluxograma B, na decisão em 6, uma coluna completa significa que a diferença entre H do espaço e a altura da coluna é menor que H da caixa (ou seja, não é possível empilhar mais uma caixa na coluna, já que passaria do teto do contêiner). Para que se carregue mais de uma coluna na iteração, é necessário que todas sejam completas, o que também significa que todas tem a mesma altura, formando assim a parede. Uma coluna incompleta (a que não atingiu a altura para ser completa) não pode ser carregada ao lado de outras, sejam elas completas ou não, na mesma iteração.

Seguindo em B, os itens 7, 9 e 10 tratam da largura flexível, citada anteriormente. Caso esta tenha sido calculada, limitamos o carregamento até seu limite, mas a última coluna pode ultrapassá-lo se estiver apenas parcialmente além dele, a Figura 8 ilustra esse caso. A justificativa para esse procedimento é que cria-se com ele uma oportunidade de, numa próxima iteração, aproveitar, por meio de uma junção, parte do espaço rejeitado deixado para trás.

O status do tipo de caixa carregado é atualizado de *”não-open”* para *”open”* na etapa 12 do fluxograma B. Feito o carregamento, a parede retangular de caixas é formada, e a partir dela ocorre a separação dos espaços, já explicada na

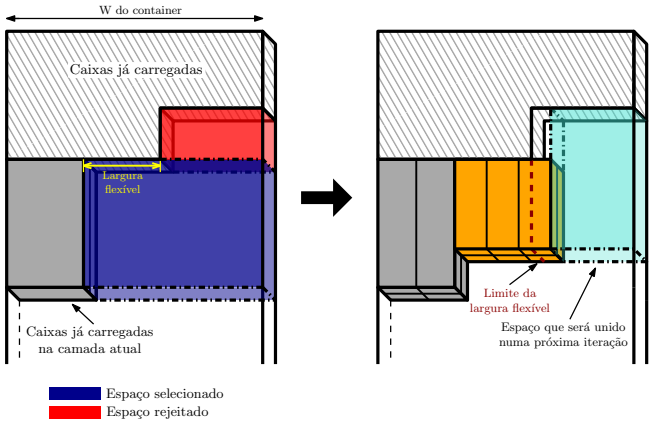

Fig. 8: Carregamento com largura flexível

Subseção III-A. Em 16 e 19, a dimensão mínima de caixa se refere a menor de todas as dimensões de todos os tipos de caixa que ainda não foram totalmente carregados. Caso a largura (W) do Espaço W ou a altura (H) do Espaço H sejam menores que essa mínima dimensão de caixa, não os adicionamos nem na lista de espaços nem na de espaços rejeitados, já que seria impossível adicionar qualquer nova caixa neles, mesmo se fosse feita uma junção de espaços. Por fim, após os itens 19 e 20, retornamos ao fluxograma A, onde é iniciada uma nova iteração se não estiverem satisfeitos os critérios de parada.

C. Modificações na heurística de G&R

Nessa seção, explicaremos duas mudanças propostas na heurística de G&R, e que assim geraram novas heurísticas. As duas modificações ocorrem na etapa 3 do fluxograma A e envolvem a lista de espaços, que sempre se altera a cada iteração com a adição e remoção de itens.

No algoritmo original, o espaço a ser selecionado da lista é sempre o último. Na primeira modificação proposta, selecionaremos o primeiro item da lista ao invés do último, e na segunda modificação, um item aleatório será escolhido. Esta última modificação, diferentemente da primeira ou da heurística original, adiciona uma componente estocástica ao método original, possibilitando a geração de diferentes soluções a cada nova reinicialização do algoritmo.

Com a segunda modificação, então, criamos uma heurística *multi-start*, como mostrado no pseudocódigo 1. No algoritmo, a melhor solução é armazenada em s^* e as soluções geradas a cada iteração em s, a função z calcula o volume ocupado pelas caixas no contêiner.

Algorithm 1 Multi-start G&R

```
s* = null
z(s*) = −∞
k = 0
repeat
    s = GR_ALTERADO()
    if z(s) > z(s*) then
        s* = s
    end if
    k = k + 1
until k = 100
return s_best                    ▷ Melhor s deve ser armazenada
```

para um mesmo problema a cada vez que é aplicada e, por isso, além de alterar a seleção do espaço, iremos resolver o mesmo problema 100 vezes com essa nova heurística, e o melhor resultado obtido é registrado. Note que nas duas, passamos agora a ter a possibilidade de começar uma nova camada sem preencher completamente a última.

IV. Resultados

A heurística original de G&R e as duas contendo as modificações propostas foram implementadas em VBA. Por conveniência, intitulamos como método 1 a heurística original, método 2 a que seleciona o primeiro espaço e método 3 a que seleciona um espaço aleatório da lista. Os dados gerados no trabalho de Bischoff e Ratcliff [40] foram utilizados para testar os três métodos.

Visando construir, com um procedimento sistemático e replicável, um grande conjunto de dados provido de variabilidade e que pudesse ser usado para comparar diferentes métodos de resolução do CLP, Bischoff e Ratcliff [40] geraram e disponibilizaram 700 exemplos de problemas de carregamento, que são divididos em 7 conjuntos. Esses dados, podem ser encontrados online em https://people.brunel.ac.uk/ mastjjb/jeb/orlib/thpackinfo.html.

Cada conjunto possui 100 problemas e é nomeado como "*thpack*", que abreviaremos apenas como "th", seguido de um número de 1 a 7. O que os difere é a quantidade de tipos de caixa presente em seus problemas (todos os problemas de um mesmo conjunto possuem o mesmo número de tipos de caixa). Seguindo de th1 até th7, a quantidade de tipos de caixa em cada conjunto é, respectivamente: 3, 5, 8, 10, 12, 15 e 20. O aumento desse número também acarreta no aumento de complexidade do problema.

As dimensões e quantidade de cada tipo de caixa são fornecidos pelos dados, bem como referências de orientação vertical, que não se encaixam nesse trabalho, e as dimensões do contêiner que se mantém as mesmas nos 700 problemas e tem os valores, em cm, de (L, W, H) = (587, 233, 220).

Em cada um dos 700 problemas, os três métodos foram aplicados, gerando assim os dados de carregamento e a porcentagem de volume ocupado no contêiner, obtida pela divisão do volume ocupado (soma do volume de todas as caixas carregadas) pelo volume total do contêiner. A Figura 9 mostra a visão, gerada por um aplicativo de desenho 3D, de dois desses carregamentos, sendo eles o problema 77 do conjunto *th3* e o problema 16 do conjunto *th6*. Nos dois, foi utilizado o método 3.

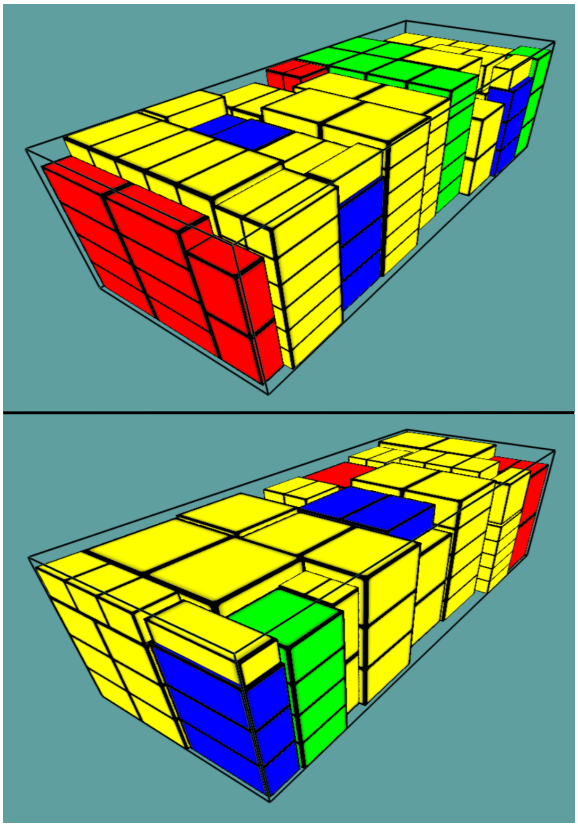

Fig. 9: Exemplos de carregamentos

Para cada um dos 7 conjuntos de 100 problemas, contamos o número de vezes em que cada um dos métodos obteve o melhor resultado entre os três. Esses valores foram então somados e e cada soma foi aplicada porcentagem sobre o conjunto total de problemas. A tabela 10 mostra esses dados.

Método / Conjunto	M1	M2	M3
th1	40	0	60
th2	7	0	93
th3	7	0	93
th4	7	0	93
th5	1	0	99
th6	4	0	96
th7	2	0	98
Total	68	0	632
%	9,71%	0%	90,29%

Fig. 10: Quantidade de melhores resultados por método

Podemos observar na Tabela 10 que o Método 3 obteve os melhores resultados na grande maioria das vezes (90,29% delas), e que essa frequência foi ainda maior nos conjuntos

com maior número de tipos de caixa. No conjunto 1, em que há apenas 3 tipos de caixa diferentes, o método 1 conseguiu 40% dos melhores resultados, ainda perdendo para o 3. Já o método 2 nenhuma vez produziu o maior volume ocupado. Na frequência de melhores resultados, comparamos também o método 2 isoladamente com o 1. A figura 11 traz esse comparativo. Nota-se que o método 1 ainda assim obteve vantagem sobre o 2 na maioria dos problemas resolvidos.

Conjunto / Método	th1	th2	th3	th4	th5	th6	th7
M1	77	66	63	74	73	68	71
M2	24	34	37	26	27	32	29

	Total	%
M1	491	70,14%
M2	209	29,86%

Fig. 11: Comparativo entre M1 e M2

Além disso, calculamos, para cada método, a média das 100 soluções encontradas (uma respectiva a cada um dos 100 problemas) por conjunto, e com isso também temos a média geral das 700 soluções, agrupando os 7 conjuntos. Esses dados estão apresentados na tabela da Figura 12 e dispostos no gráfico de linhas da Figura 13. Vemos que o método 2 obteve a menor média em todos os conjuntos, ficando abaixo da heurística original em 1,59%. Comparativamente, a diferença foi maior entre os resultados da terceira heurística e os da primeira. A média de volume ocupado usando o método 3 foi 3,38% maior que o da heurística original, o que, dado que esta já produziu uma média de ocupação de 82,77% (restando assim 17,23% de melhoria possível para um perfeito e até utópico 100%), é um valor de incremento interessante.

Método / Conjunto	M1	M2	M3
th1	84,78%	83,12%	85,68%
th2	83,18%	81,83%	86,49%
th3	83,02%	82,23%	86,92%
th4	82,95%	80,87%	86,53%
th5	82,16%	80,55%	86,26%
th6	82,08%	80,37%	85,97%
th7	81,25%	79,34%	85,18%
Média	82,77%	81,18%	86,15%

Fig. 12: Porcentagem de volume ocupado com cada método

V. Conclusão

Este trabalho tratou do problema de Carregamento de Contêineres. Foi apresentada uma releitura da conhecida heurística de G&R. As etapas e características desta foram

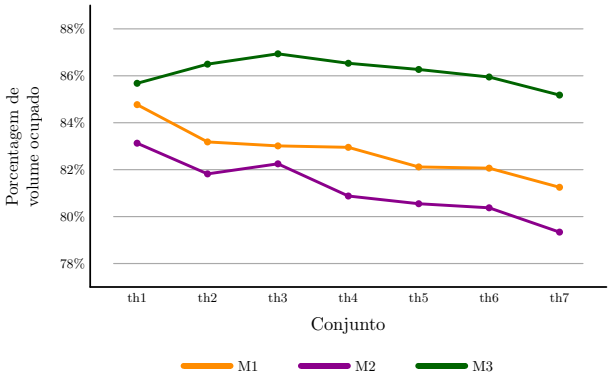

Fig. 13: Comparativo de volumes ocupados

unificadas por meio de dois fluxogramas intuitivos e ilustradas com novas imagens, não fornecidas no artigo original. Além disso, foram propostas duas modificações na heurística, especificamente na etapa em que é selecionado, na lista de espaços, aquele que será trabalhado em cada iteração. No método original, o último espaço da lista é selecionado, enquanto, no método 2, é o primeiro e, no método 3, é escolhido um espaço aleatório da lista e a heurística toda é aplicada 100 vezes para o mesmo problema, selecionando o melhor resultado encontrado como solução.

Os três métodos foram implementados em VBA e testados computacionalmente utilizando um extenso banco de dados com 700 problemas de carregamento, fornecido no trabalho de Bischoff e Ratcliff [40]. Os três métodos levaram um tempo computacional pequeno para gerar soluções. O método 2 não conseguiu superar o método 1 (a heurística original), obteve resultados melhores ocasionalmente, mas na maioria das vezes gerou um volume de ocupação menor, o que se refletiu em não ter conseguido uma ocupação média maior em nenhum dos conjuntos de problemas.

O método 3, por sua vez, conseguiu se sobressair em relação à heurística original. Em todos os conjuntos de problemas, encontrou uma melhor solução na maioria das vezes e conseguiu gerar um maior volume ocupado. No conjunto th1, a diferença foi pouco expressiva, mas nos outros 6, o método 3 gerou uma melhor solução quase que na totalidade das vezes e aumentou o volume ocupado em uma porcentagem interessante.

Por fim, esse trabalho oferece um bom material para se compreender não só a heurística de G&R, uma das pioneiras a tratar do Problema de Carregamento de Contêineres, mas também vários conceitos que foram introduzidos por esses autores e posteriormente utilizados em diversos outros métodos propostos para resolver o problema. Além disso, os resultados encontrados aqui fornecem uma solução prática para ser aplicada em empresas que trabalham com carregamento de contêineres e uma oportunidade para pesquisas na área trabalharem com um novo método heurístico de se chegar numa solução. Para pesquisas futuras, uma oportunidade a ser explorada é o uso do método proposto aqui na geração de uma solução inicial na qual seriam então utilizadas meta-heurísticas para melhorar ainda mais a solução.

REFERENCES

[1] D. Waters, *Logistics An Introduction to supply chain management*. Palgrave macmillan, 2021.

[2] G. Ghiani, G. Laporte, and R. Musmanno, *Introduction to logistics systems planning and control*. John Wiley & Sons, 2004.

[3] Y. Huang, L. Lai, W. Li, and H. Wang, "A differential evolution algorithm with ternary search tree for solving the three-dimensional packing problem," *Information Sciences*, vol. 606, pp. 440–452, 2022.

[4] R. Ranck Jr, H. H. Yanasse, and R. Morabito, "Contribuições para um problema de carregamento de contêiner com múltiplos compartimentos."

[5] J. A. George and D. F. Robinson, "A heuristic for packing boxes into a container," *Computers & Operations Research*, vol. 7, no. 3, pp. 147–156, 1980.

[6] F. O. Ceccilio and R. Morabito, "Refinamentos na heurística de george e robinson para o problema do carregamento de caixas dentro de contêineres," *Transportes*, vol. 12, no. 1, 2004.

[7] M. Delorme and J. Wagenaar, "Exact decomposition approaches for a single container loading problem with stacking constraints and medium-sized weakly heterogeneous items," *Omega*, vol. 125, p. 103039, 2024.

[8] C.-S. Chen, S.-M. Lee, and Q. Shen, "An analytical model for the container loading problem," *European Journal of operational research*, vol. 80, no. 1, pp. 68–76, 1995.

[9] M. Hifi, I. Kacem, S. Nègre, and L. Wu, "A linear programming approach for the three-dimensional bin-packing problem," *Electronic Notes in Discrete Mathematics*, vol. 36, pp. 993–1000, 2010.

[10] J.-F. Tsai, P.-C. Wang, and M.-H. Lin, "A global optimization approach for solving three-dimensional open dimension rectangular packing problems," *Optimization*, vol. 64, no. 12, pp. 2601–2618, 2015.

[11] C. Paquay, M. Schyns, and S. Limbourg, "A mixed integer programming formulation for the three-dimensional bin packing problem deriving from an air cargo application," *International Transactions in Operational Research*, vol. 23, no. 1-2, pp. 187–213, 2016.

[12] G. Fasano, "A mip approach for some practical packing problems: Balancing constraints and tetris-like items," *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, vol. 2, no. 2, pp. 161–174, 2004.

[13] L. Junqueira, R. Morabito, and D. S. Yamashita, "Three-dimensional container loading models with cargo stability and load bearing constraints," *Computers & Operations Research*, vol. 39, no. 1, pp. 74–85, 2012.

[14] S. Martello, D. Pisinger, and D. Vigo, "The three-dimensional bin packing problem," *Operations research*, vol. 48, no. 2, pp. 256–267, 2000.

[15] I. Araya, M. Moyano, and C. Sanchez, "A beam search algorithm for the biobjective container loading problem," *European Journal of Operational Research*, vol. 286, no. 2, pp. 417–431, 2020.

[16] N. Wang, A. Lim, and W. Zhu, "A multi-round partial beam search approach for the single container loading problem with shipment priority," *International Journal of Production Economics*, vol. 145, no. 2, pp. 531–540, 2013.

[17] A. G. Ramos, E. Silva, and J. F. Oliveira, "A new load balance methodology for container loading problem in road transportation," *European Journal of Operational Research*, vol. 266, no. 3, pp. 1140–1152, 2018.

[18] A. Bortfeldt and H. Gehring, "A hybrid genetic algorithm for the container loading problem," *European Journal of Operational Research*, vol. 131, no. 1, pp. 143–161, 2001.

[19] T. Jamrus and C.-F. Chien, "Extended priority-based hybrid genetic algorithm for the less-than-container loading problem," *Computers & Industrial Engineering*, vol. 96, pp. 227–236, 2016.

[20] J.-N. Zheng, C.-F. Chien, and M. Gen, "Multi-objective multi-population biased random-key genetic algorithm for the 3-d container loading problem," *Computers & Industrial Engineering*, vol. 89, pp. 80–87, 2015.

[21] T. Bayraktar, F. Ersöz, and C. Kubat, "Effects of memory and genetic operators on artificial bee colony algorithm for single container loading problem," *Applied Soft Computing*, vol. 108, p. 107462, 2021.

[22] T. Dereli and G. Sena Das, "A hybrid simulated annealing algorithm for solving multi-objective container-loading problems," *Applied Artificial Intelligence*, vol. 24, no. 5, pp. 463–486, 2010.

[23] H. Mostaghimi Ghomi, B. G. St Amour, and W. Abdul-Kader, "Three-dimensional container loading: A simulated annealing approach," *International Journal of Applied Engineering Research*, vol. 12, no. 7, p. 1290, 2017.

[24] L. Sheng, S. Xiuqin, C. Changjian, Z. Hongxia, S. Dayong, and W. Feiyue, "Heuristic algorithm for the container loading problem with multiple constraints," *Computers & Industrial Engineering*, vol. 108, pp. 149–164, 2017.

[25] D. Mack, A. Bortfeldt, and H. Gehring, "A parallel hybrid local search algorithm for the container loading problem," *International Transactions in Operational Research*, vol. 11, no. 5, pp. 511–533, 2004.

[26] A. Bortfeldt, H. Gehring, and D. Mack, "A parallel tabu search algorithm for solving the container loading problem," *Parallel computing*, vol. 29, no. 5, pp. 641–662, 2003.

[27] M. Gendreau, M. Iori, G. Laporte, and S. Martello, "A tabu search algorithm for a routing and container loading problem," *Transportation Science*, vol. 40, no. 3, pp. 342–350, 2006.

[28] C. D. Tarantilis, E. E. Zachariadis, and C. T. Kiranoudis, "A hybrid metaheuristic algorithm for the integrated vehicle routing and three-dimensional container-loading problem," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 2, pp. 255–271, 2009.

[29] J. Liu, Y. Yue, Z. Dong, C. Maple, and M. Keech, "A novel hybrid tabu search approach to container loading," *Computers & Operations Research*, vol. 38, no. 4, pp. 797–807, 2011.

[30] E. E. Bischoff and M. D. Marriott, "A comparative evaluation of heuristics for container loading," *European Journal of Operational Research*, vol. 44, no. 2, pp. 267–276, 1990.

[31] A. Moura and J. F. Oliveira, "A grasp approach to the container-loading problem," *IEEE Intelligent Systems*, vol. 20, no. 4, pp. 50–57, 2005.

[32] H. Gehring *et al.*, "A genetic algorithm for solving the container loading problem," *International transactions in operational research*, vol. 4, no. 5-6, pp. 401–418, 1997.

[33] M. Eley, "Solving container loading problems by block arrangement," *European Journal of Operational Research*, vol. 141, no. 2, pp. 393–409, 2002.

[34] T. Fanslau and A. Bortfeldt, "A tree search algorithm for solving the container loading problem," *INFORMS Journal on Computing*, vol. 22, no. 2, pp. 222–235, 2010.

[35] W. Zhu, W.-C. Oon, A. Lim, and Y. Weng, "The six elements to block-building approaches for the single container loading problem," *Applied Intelligence*, vol. 37, pp. 431–445, 2012.

[36] D. Zhang, Y. Peng, and S. C. Leung, "A heuristic block-loading algorithm based on multi-layer search for the container loading problem," *Computers & Operations Research*, vol. 39, no. 10, pp. 2267–2276, 2012.

[37] R. Morabito and M. Arenales, "An and/or-graph approach to the container loading problem," *International Transactions in Operational Research*, vol. 1, no. 1, pp. 59–73, 1994.

[38] B. Ngoi, M. Tay, and E. Chua, "Applying spatial representation techniques to the container packing problem," *The International Journal of Production Research*, vol. 32, no. 1, pp. 111–123, 1994.

[39] W. Huang and K. He, "A caving degree approach for the single container loading problem," *European Journal of Operational Research*, vol. 196, no. 1, pp. 93–101, 2009.

[40] E. E. Bischoff and M. Ratcliff, "Issues in the development of approaches to container loading," *Omega*, vol. 23, no. 4, pp. 377–390, 1995.