

ALGORITMO DA COLÔNIA DE FORMIGAS PARA O PROBLEMA DO EMPACOTAMENTO TRIDIMENSIONAL COM ROTAÇÃO

Daniel Bento Maia

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul
R. Gen. Osório, 348, Centro, Bento Gonçalves, RS, 95700-086
daniel.maia@ifrs.edu.br

Denis Borenstein

Escola de Administração, Universidade Federal do Rio Grande do Sul
Rua Washington Luiz, 855, Centro, Porto Alegre, RS, 90010-460
denis.borenstein@ufrgs.br

Luciano Ferreira

Escola de Administração, Universidade Federal do Rio Grande do Sul
Rua Washington Luiz, 855, Centro, Porto Alegre, RS, 90010-460
ferreira.luciano@ufrgs.br

ABSTRACT

Este trabalho propõe um método rápido para resolver o problema de empacotamento tridimensional (3D-BPP) com rotação de itens utilizando o Algoritmo de Otimização da Colônia de Formigas. O 3D-BPP é um problema clássico de otimização combinatória considerado NP-difícil. O método desenvolvido apresenta diversas características que permitem sua adaptação eficiente a diferentes contextos, incluindo desde instâncias de pequeno porte até aquelas de grande escala, além de lidar bem com itens bastante heterogêneos. A eficácia da proposta foi avaliada por meio de instâncias de teste amplamente utilizadas na literatura, envolvendo diversas combinações de tamanhos de caixas e itens. Apesar de não atingir os melhores resultados conhecidos em todas as instâncias, o algoritmo demonstrou desempenho competitivo de forma consistente, com tempos de execução consideravelmente inferiores aos métodos já existentes.

KEYWORDS. Problema do Empacotamento Tridimensional. Problema do Carregamento de Contêiner. Algoritmo de Otimização da Colônia de Formigas.

ABSTRACT

This study proposes a fast method for solving the three-dimensional bin packing problem (3D-BPP) with item rotation using the Ant Colony Optimization algorithm. The 3D-BPP is a classic combinatorial optimization problem classified as NP-hard. The developed method incorporates several features that enable efficient adaptation to different scenarios, ranging from small to large instances, and effectively handles highly heterogeneous items. The effectiveness of the proposed approach was assessed using widely adopted benchmark instances from the literature, involving various combinations of bin and item sizes. Although it did not achieve the best-known results for all instances, the algorithm consistently delivered competitive performance with significantly shorter execution times than existing methods.

KEYWORDS. 3D Bin Packing Problem. Container loading problem. Ant colony optimization.

Eixos Temáticos: L&T - Logística e Transporte. MH - Metaheurística. OD - Otimização Discreta.

1. Introdução

O Problema de Empacotamento Tridimensional (3D-BPP), ou o Problema de Carregamento com Múltiplos Contêineres (m-CLP), visa empacotar de forma eficiente itens retangulares diversos em compartimentos ou contêineres de dimensões homogêneas, minimizando o número de compartimentos utilizados. Este problema pertence à classe do Problema de Empacotamento com Compartimento de Tamanho Único (SBSBPP), conforme definido por Wäscher et al. [2007]. O foco está em uma variante do 3D-BPP caracterizada pelas seguintes condições: (i) os contêineres possuem tamanho uniforme; (ii) os itens podem ser rotacionados e apresentam uma ampla variedade de heterogeneidade, de fraca a fortemente heterogêneos; e (iii) as únicas restrições são os itens não ultrapassarem os limites do contêiner e os itens não se sobreporem entre si. Conforme estabelecido por Pisinger [2002], o problema de carregamento de contêineres é NP-difícil e, consequentemente, o 3D-BPP herda essa complexidade computacional.

Diversos métodos têm sido propostos na literatura para resolver o 3D-BPP/m-CLP com rotação de itens. Gonçalves and Resende [2013] desenvolveram um algoritmo genético com chave aleatória enviesada (BRKGA) para resolução de problemas de empacotamento 2D e 3D com rotação. Esse algoritmo introduziu duas novas heurísticas e uma função de aptidão eficiente, obtendo resultados competitivos em comparação com métodos anteriores sem rotação. Feng et al. [2015] propuseram dois algoritmos genéticos híbridos (HGAs) para lidar com problemas tanto de pequena quanto de grande escala com rotação, especialmente em cenários fortemente heterogêneos. Embora esses HGAs tenham demonstrado desempenho robusto, seu elevado tempo de CPU limitou sua viabilidade para aplicações em tempo real. Mahvash et al. [2017] desenvolveram uma heurística baseada em geração de colunas (CG) incorporando rotação. Sua estrutura branch-and-price superou algoritmos anteriores no tratamento de itens rotacionados. Mais recentemente, Şafak and Erdoğan [2023] propuseram um algoritmo de busca em grande vizinhança para o m-CLP, abordando restrições práticas adicionais como orientação, empilhamento, limites de peso, exigências de manuseio e estabilidade. No entanto, o método parece otimizado para casos de heterogeneidade fraca, com variação mínima nas dimensões dos itens.

Contudo, essas abordagens frequentemente exigem tempo computacional elevado para resolver instâncias de referência, limitando sua aplicabilidade em contextos reais. A crescente prevalência de ambientes de tomada de decisão *online* [Ali et al., 2022] e o surgimento de problemas integrados de empacotamento multidimensional com roteamento e agendamento de veículos [Liu et al., 2024] evidenciam a necessidade de métodos *offline* altamente eficientes, que possam servir como componentes fundamentais nesses contextos mais complexos e dinâmicos.

O presente estudo propõe uma abordagem baseada no Algoritmo de Colônia de Formigas (ACO) para resolver o 3D-BPP/m-CLP com rotação de itens. O algoritmo ACO explora eficientemente o espaço de soluções utilizando uma heurística construtiva, o método do Ponto Extremo (EP) [Crainic et al., 2008]. A integração das heurísticas EP dentro da estrutura ACO permite tanto a diversificação — explorando novas regiões do espaço de soluções — quanto a intensificação — aprofundando a busca em espaços promissores.

Experimentos computacionais foram conduzidos em instâncias pequenas e grandes, caracterizadas por conjuntos de itens fortemente heterogêneos. O desempenho do algoritmo proposto foi comparado com os métodos mais avançados reportados na literatura, abrangendo um conjunto diverso de instâncias de problema. Os resultados demonstram que a abordagem proposta apresenta desempenho competitivo em termos de eficiência e eficácia. O procedimento desenvolvido oferece uma base promissora para o desenvolvimento futuro de algoritmos, especialmente para aplicações que integrem empacotamento multidimensional com exigências logísticas reais, como roteamento e agendamento de veículos. O método também pode ser incorporado a outras metodologias como

uma gerador de um conjunto de boas soluções iniciais.

O restante do artigo está organizado da seguinte forma. A Seção 2 descreve a formulação formal do problema. A Seção 3 detalha o algoritmo ACO. A Seção 4 apresenta os experimentos computacionais e uma análise comparativa resultados prévios. Por fim, a Seção 5 conclui o artigo e sugere direções para pesquisas futuras.

2. Formulação do modelo

A estrutura conceitual para a modelagem do 3D-BPP baseia-se nas contribuições de trabalhos anteriores [Chen et al., 1995; Mahvash et al., 2017]. Cada item cúbico $i \in N$ é caracterizado por suas dimensões (l_i, w_i, h_i) , representando a orientação inicial do item paralela ao eixo x , y e z . Simultaneamente, cada contêiner $j \in B$ destinado ao empacotamento é posicionado com seu canto inferior-traseiro-esquerdo no ponto $(0, 0, 0)$. Os limites de um contêiner j ao longo dos eixos x , y e z são denotados pelos pontos L_j, W_j, H_j . Após o posicionamento dentro de um contêiner j , as coordenadas (x_i, y_i, z_i) são variáveis de decisão que definem a localização do vértice inferior-traseiro-esquerdo do item i .

Os itens podem assumir seis orientações, conforme Mahvash et al. [2017], representadas pelas variáveis binárias $(o_{i1}, o_{i2}, o_{i3}, o_{i4}, o_{i5}, o_{i6})$. Essas variáveis assumem o valor 1 se o item i estiver na orientação indicada na Tabela 1, e 0, caso contrário.

Tabela 1: Seis rotações

Orientação	Variável	Projeções		
		x	y	z
1	o_{i1}	l	w	h
2	o_{i2}	w	l	h
3	o_{i3}	h	w	l
4	o_{i4}	w	h	l
5	o_{i5}	l	h	w
6	o_{i6}	h	l	w

Outra representação importante envolve capturar as posições relativas entre os itens, as quais são expressas por meio de variáveis binárias denotadas como $a_{ik}, b_{ik}, c_{ik}, d_{ik}, e_{ik}$ e f_{ik} . Essas variáveis binárias assumem o valor 1 quando o item i está posicionado atrás, à frente, à esquerda, à direita, abaixo e acima do item k , respectivamente. Além disso, as variáveis a_{ik}, c_{ik} e e_{ik} indicam que o item i está posicionado antes do item k em relação aos eixos x , y e z , respectivamente. Por outro lado, as variáveis b_{ik}, d_{ik} e f_{ik} indicam que o item k está posicionado antes do item i ao longo dos eixos x , y e z .

Adicionalmente, introduzimos as variáveis s_{ij} , que indicam se o item i está carregado no contêiner j , e a variável binária n_j , com $n_j = 1$ se o contêiner $j \in B$ for utilizado, e $n_j = 0$, caso contrário. Com base na notação apresentada acima, utilizamos a formulação de Mahvash et al. [2017] para modelar o problema. Por questões economia de espaço, não apresentamos esta formulação. O objetivo da formulação é minimizar o número de compartimentos/contêineres utilizados. A formulação contém restrições que: (i) garantem que as dimensões dos itens carregados não ultrapassem as dimensões do contêiner; (ii) proíbem sobreposição entre itens carregados; (iii) especificam que cada item seja orientado de apenas uma maneira; (iv) garantem que os itens carregados apresentem posições relativas permitidas entre si; (v) asseguram que um item seja alocado em apenas um contêiner; e (vi) garantam que os contêineres com pelo menos um item carregado sejam efetivamente utilizados. Dada a natureza NP-difícil do 3D-BPP, propomos um método heurístico para tratar o problema de forma eficiente e eficaz.

3. Algoritmo de Otimização da Colônia de Formigas

O Algoritmo de Otimização de Colônia de Formigas (ACO) é um algoritmo de aprendizado por reforço utilizado para resolver problemas combinatórios complexos Liu et al. [2021]. O

ACO envolve um processo iterativo que gera soluções viáveis adicionais com base no rastro de feromônio deixado por soluções viáveis bem-sucedidas identificadas durante o processo de busca. A abordagem é inspirada no comportamento real das formigas na busca por alimentos. Inicialmente, as formigas exploram o ambiente aleatoriamente e, ao encontrarem alimento, deixam um rastro de feromônio cuja intensidade depende da qualidade da descoberta.

Nossa abordagem define dois tipos distintos de feromônios. Os feromônios de itens representam a ordem em que os itens são carregados nos contêineres. O feromônio de combinação representa diversas combinações dos seguintes três parâmetros: critério de ordenação dos itens, regras de rotação e critérios de ordenação dos espaços residuais. O Algoritmo EP utiliza esses parâmetros para empacotar os itens nos contêineres. O conjunto de formigas é representado por ANT , com a cardinalidade determinada pelo tamanho do conjunto de itens $|N|$ multiplicado pelo parâmetro Q , que varia entre zero e um. Vale destacar que os feromônios de itens também dependem da combinação utilizada. Cada item tem uma quantidade de feromônio depositada para cada combinação (τ_i^c).

A necessidade de introduzir combinações de múltiplas prioridades de aplicação das regras de rotação e critérios de utilização do espaço residual surge da necessidade de fornecer flexibilidade à nossa abordagem de otimização. As combinações permitem melhor aproveitamento do espaço dos contêineres, considerando diferentes situações de quantidades e dimensões de itens. Embora essas combinações aumentem a complexidade do algoritmo, o desempenho geral não é significativamente impactado. Como diversas possibilidades são testadas em cada iteração, o algoritmo encontra boas soluções mais rapidamente, exigindo menos iterações no total.

Nossa abordagem ACO baseia-se no desenvolvimento de dois algoritmos aplicados em sequência para lidar com esses dois tipos de feromônio. O primeiro algoritmo (Algoritmo 1) trata apenas dos feromônios de combinação, e seu principal objetivo é identificar o melhor conjunto de combinações para um problema específico, representado pela quantidade de feromônio para cada combinação, τ_c . O segundo algoritmo (Algoritmo 2) lida com os dois tipos de feromônios para encontrar boas soluções para o problema, sendo uma versão customizada para o 3D-BPP da metaheurística apresentada em Dorigo and Blum [2005].

Vamos focar no primeiro algoritmo, a Inicialização de Combinação, apresentado no Algoritmo 1. Além de identificar e ranquear as melhores combinações de parâmetros para o empacotamento de itens utilizando o algoritmo EP, este algoritmo também determina e armazena as boas soluções associadas às melhores combinações. O algoritmo utiliza a seguinte notação adicional: s é uma solução viável; $f'(s)$ é o valor da solução s , calculado neste algoritmo como a função objetivo de s acrescida da porcentagem de volume utilizado do contêiner mais ocioso; S_{col} é um conjunto de soluções viáveis; e $LB = \frac{\sum_{i \in N} v_i}{V}$ é o limite inferior contínuo, onde v_i é o volume do item i e V é o volume do contêiner.

O algoritmo EP (Algoritmo desenvolvido por Crainic et al. [2008]) é chamado diversas vezes pelo Algoritmo 1. Esse algoritmo tem como objetivo gerar padrões de empacotamento viáveis para os algoritmos ACO. Como estamos desenvolvendo um modelo capaz de lidar com várias variantes no contexto do 3D-BPP, a questão da rotação precisou ser mais aprofundada do que em trabalhos anteriores [Feng et al., 2015; Mahvash et al., 2017].

Aplicamos/desenvolvemos três regras de rotação. A Regra de Rotação 1 (RR1) é baseada em Mahvash et al. [2017], que inicialmente define a projeção de um dos lados do item i sobre um lado do espaço residual, com dimensões (L_{rs}, W_{rs}, H_{rs}) , que apresente o menor valor não negativo considerando as expressões: $(L_{rs} - l_i), (L_{rs} - w_i), (L_{rs} - h_i), (W_{rs} - l_i), (W_{rs} - w_i), (W_{rs} - h_i), (H_{rs} - l_i), (H_{rs} - w_i), H_{rs} - h_i$. Após a escolha, são excluídos do conjunto os elementos que contêm o eixo do espaço residual ou o lado do item utilizados. Então, repete-se o passo anterior

Algorithm 1 Algoritmo de Iniciação das Combinações

Input: Sets ON , B , and C , LB

```

 $s_{bs} \leftarrow \text{NULL}$ 
 $S_{col} \leftarrow \emptyset$ 
 $\tau_c \leftarrow 0, \forall c \in C$ 
for each  $j \in C$  do
     $ON \leftarrow \text{ordenaItem}(N, c_{j1})$ 
     $s \leftarrow \text{EP}(ON, B, c_{j2}, c_{j3})$ 
     $\text{updateCombinationPheromone}(c, LB, f'(s), \tau_c)$ 
     $S_{col} \leftarrow S_{col} \cup \{s\}$ 
    if  $((s_{bs} = \text{NULL}) \text{ or } (f'(s) < f'(s_{bs})))$  then
         $s_{bs} \leftarrow s$ 
    end if
end for

```

Output: s_{bs} , S_{col} , τ_c

para definir a segunda projeção referente a orientação do item.

Nosso estudo incorporou duas novas regras de rotação adicionais que dependem das dimensões do contêiner. Os itens são rotacionados com base em dois movimentos principais ao longo dos eixos do contêiner. O movimento prioritário é projetar o lado mais longo do item (LS_i) paralelamente a um dos eixos do contêiner e depois projetar o segundo maior lado do item (SLS_i) paralelamente a outro eixo do contêiner. A Tabela 2 apresenta os possíveis movimentos de rotação com base nas dimensões do contêiner.

Tabela 2: Movimentos de Rotação

Rotação	Dimensões do Contêiner		
	Maior Lado	Segundo Maior Lado	Menor Lado
1	–	SLS_i	LS_i
2	–	LS_i	SLS_i
3	LS_i	SLS_i	–
4	LS_i	–	SLS_i
5	SLS_i	LS_i	–
6	SLS_i	–	LS_i

A Regra de Rotação 2 (RR2) começa na linha 1 da Tabela 2 e vai até a linha 6, parando na primeira opção em que seja possível acomodar o item. A Regra de Rotação 3 (RR3) começa na linha 6 da Tabela 2, seguindo para a linha 1, e vai até a linha 5, parando na primeira opção em que seja possível acomodar o item.

Além disso, introduzimos novas formas de lidar com os espaços residuais. Em Crainic et al. [2008], apenas os critérios de ordenação dos itens foram avaliados. No entanto, a eficácia do algoritmo EP pode ser significativamente aprimorada quando os espaços residuais também são ordenados, especialmente ao considerar diferentes níveis de heterogeneidade dos itens. Os três seguintes critérios de ordenação de espaços residuais foram implementados no procedimento *PutItemInBin*: (i) RSSC 1: Menor Volume; (ii) RSSC 2: Espaço Residual com a menor coordenada no eixo do lado mais longo do contêiner, 1º critério de desempate: menor coordenada no lado menor do contêiner, 2º critério de desempate: menor coordenada no eixo ainda não verificado; e (iii) RSSC 3: Espaço Residual com a menor coordenada no eixo do lado maior do contêiner, 1º critério de desempate: menor coordenada no segundo maior lado do contêiner, 2º critério de desempate: menor coordenada no eixo menor do contêiner.

Nossa abordagem executa o Algoritmo EP com três parâmetros: critério de ordenação dos itens, regras de rotação e critérios de ordenação dos espaços residuais. Quatro critérios dife-

Tabela 3: Parâmetros de combinações para utilização no Algoritmo EP 2

Combinação	Critério de ordenamento dos itens	Regra de Rotação	Critério de Ordenamento dos EPs
c_1	ISC1	RR1	RSSC1
c_2	ISC1	RR2	RSSC1
c_3	ISC2	RR3	RSSC2
c_4	ISC2	RR3	RSSC3
c_5	ISC3	RR1	RSSC1
c_6	ISC4	RR1	RSSC1
c_7	ISC4	RR2	RSSC1

rentes são usados para a ordenação dos itens no Algoritmo 1, conforme a seguir: (i) ISC1: Ordem decrescente do volume dos itens; (ii) ISC2: Ordem decrescente do maior lado dos itens; (iii) ISC3: Ordem decrescente do menor lado dos itens; e (iv) ISC4: Ordem decrescente da menor área dos itens. Todas as combinações dos quatro critérios de ordenação dos itens, das três regras de rotação e dos três critérios de ordenação dos espaços residuais foram avaliadas por meio de experimentos extensivos. No entanto, as sete combinações apresentadas na Tabela 3 dominaram as demais regras em termos de minimização do número de contêineres necessários para empacotar os itens. Somente essas sete combinações foram utilizadas durante os experimentos numéricos descritos na Seção 4. Cada combinação da Tabela 3 é armazenada no conjunto C como uma n -pla $c_j = (ISC, RR, RSSC), j = 1, \dots, 7$.

Os feromônios da combinação são atualizados em `updateCombinationPheromone` usando a regra 1:

$$\tau_c \leftarrow \tau_c \cdot (1 - e) + \frac{LB}{f'(s)} \quad (1)$$

Após a execução do algoritmo 1, os valores de τ_c são utilizados pelo ACO (Algoritmo 2) como ponto de partida.

O algoritmo é inicializado chamando a rotina `InitializePheromoneValue` para computar a probabilidade de cada item $i \in N$ ser carregado com um valor inicial de $\tau_i^c \geq 0, \forall c, \forall i$. A probabilidade é calculada pela equação

$$p_i^c = \frac{[\tau_i^c]^\alpha [\eta(\epsilon_i^c)]^\beta}{\sum_{j \in N} [\tau_j^c]^\alpha [\eta(\epsilon_j^c)]^\beta} \quad (2)$$

onde o valor heurístico ϵ_i^c pode ser definido pela primeira posição da combinação c_j sorteada. O parâmetro α controla a influência do feromônio, enquanto os parâmetros β e η controlam a influência do valor heurístico ϵ_i^c . Estes valores ponderam os pesos dos feromônios e da intuição heurística no cálculo da probabilidade. Além disto, $\eta = Q_\eta \max_{i \in N} V_i$, onde Q_η é um parâmetro definido entre 0 e 1.

Nas rotinas `calculateProbabilityCombination` e `calculateProbabilityOrder`, as probabilidades para cada item e combinações são atualizadas baseadas nos valores de τ_c e τ_i^c computados na iteração anterior. A probabilidade dos itens são calculadas na Eq. (2), enquanto a probabilidade de cada combinação é calculada como se segue:

$$p_c = \frac{\tau_c}{\sum_{c \in C} \tau_c}$$

No procedimento `rouletteWheel`, os itens são ordenados no conjunto ON utilizando um método padrão de roleta (*roulette wheel selection*), no qual os itens com maior probabilidade têm mais chances de serem selecionados primeiro. Os itens em ON são empacotados usando o

Algorithm 2 Algoritmo de Otimização de Colônia de Formigas

Input: Sets N , B , and C , and LB

```

 $ANT \leftarrow Q|N|$ ,  $totalTime \leftarrow 0$ 
InitializePheromoneValue( $\tau_i^c$ ),  $\forall c \in C, \forall i \in N$ 
 $limitImprovement \leftarrow 0.15 * timeLimit$ ,  $timeLastImprovement \leftarrow 0$ 
while ( $totalTime \leq limitTime$ ) or ( $timeLastImprovement < limitImprovement$ ) do
     $p_c \leftarrow calculateProbabilityCombination(\tau_c, c)$ ,  $\forall c \in C$ 
     $p_{ci} \leftarrow calculateProbabilityOrder(\tau_i^c, c)$ ,  $\forall c \in C, \forall i \in N$ 
     $s_{iter} \leftarrow NULL$ 
     $S_{col} \leftarrow \emptyset$ 
    for  $j = 1, \dots, ANT$  do
         $c_j \leftarrow rouletteWheelComb(p_c)$ 
         $ON \leftarrow rouletteWheelItem(p_i^c, c_j)$ 
         $s \leftarrow EP(ON, B, c_j)$ 
        if (( $s_{iter} = NULL$ ) or ( $f'(s) < f'(s_{iter})$ )) then
             $s_{iter} \leftarrow s$ 
             $c_{iter} \leftarrow c_j$ 
             $ON_{iter} \leftarrow ON$ 
        end if
    end for
     $S_{col} \leftarrow S_{col} \cup \{s_{iter}\}$ 
    updateCombinationPheromone( $c_{iter}$ ,  $LB$ ,  $s_{iter}$ ,  $\tau_c$ )
    updateItemPheromone( $c_{iter}.LB$ ,  $s_{iter}$ ,  $\tau_i^c$ ,  $ON_{iter}$ )
end while
 $s_{bs} \leftarrow \min_{S_{col}} \{f'(s)\}$ 
updateTimeLastImprovement()

```

Output: s_{bs} , S_{col}

algoritmo EP para encontrar uma solução viável, seguindo os parâmetros definidos nas combinações apresentadas na Tabela 3.

Após a conclusão de ANT iterações, a melhor solução da iteração s_{iter} é selecionada. Essa solução é utilizada nas rotinas `updateCombinationPheromone` e `updateItemPheromone` para atualizar, respectivamente, os feromônios das combinações e dos itens. Ambos os procedimentos utilizam a melhor solução e a melhor combinação de cada iteração para atualizar os valores dos feromônios, compreendendo duas etapas: (i) evaporação e (ii) depósito de feromônio na trilha. A evaporação permite o esquecimento de trilhas que foram pouco utilizadas. Para os feromônios dos itens, utilizamos um coeficiente de evaporação e . A evaporação é calculada como $\tau_i^c = \tau_i^c(1 - e)$. O depósito de feromônio na trilha é calculado por

$$\tau_i^c = \tau_i^c + \delta - (|N| - ON[i] - 1)d + a \frac{r}{f'(s_{iter})} \quad (3)$$

onde δ é o feromônio depositado, o parâmetro d representa a porcentagem de decréscimo que é inversamente proporcional à posição do item i na melhor solução da iteração, calculado pela cardinalidade do grupo de itens ($|N|$) diminuída da posição do item i na melhor solução da iteração $ON[i]$. Além disso, o feromônio recebe um acréscimo por um termo calculado com base na qualidade da melhor solução da iteração, em comparação com o limite inferior LB , mitigado pelo parâmetro a . Na melhor solução da iteração, os itens recebem uma quantidade de feromônio inversamente proporcional à ordem de carregamento. Para os feromônios das combinações, a evaporação é obtida por $\tau_c = \tau_c(1 - e)$.

Por fim, a rotina `updateTimeLastImprovement` atualiza o tempo decorrido desde a última atualização de s_{bs} até o tempo atual.

4. Experimentos numéricos

Nossos algoritmos desenvolvidos foram programados em linguagem C. Os testes computacionais foram realizados em um computador Intel com processador AMD de 3,7GHz e 16GB de RAM.

4.1. Pequenas Instâncias

Utilizamos as instâncias de *benchmark* introduzidas por Martello et al. [2007] para testar o desempenho do nosso algoritmo em instâncias fortemente heterogêneas. As instâncias são categorizadas em oito classes, cada uma contendo quatro problemas com 50, 100, 150 e 200 itens, resultando em 320 instâncias. A Tabela 4 apresenta as dimensões dos contêineres utilizadas em cada classe. O tipo de itens a serem empacotados é gerado seguindo distribuições uniformes aleatórias com parâmetros baseados nas dimensões do contêiner ou em valores fixos. Diferentes combinações de tipos de itens são usadas em cada classe. Para cada problema, 10 instâncias são geradas aleatoriamente. Mahvash et al. [2017] explica detalhadamente como as dimensões dos contêineres e dos itens são geradas para cada classe. As instâncias podem ser geradas usando um código em C disponível em <http://hjemmesider.diku.dk/~pisinger/codes.html>.

Comparamos várias implementações da nossa abordagem proposta, utilizando os resultados obtidos por Mahvash et al. [2017] como referência. ACO é o algoritmo de colônia de formigas (Algoritmo 2). Como *benchmarks*, utilizamos a abordagem de geração de colunas de Mahvash et al. [2017] (M-CG), o algoritmo BRKGA com seis rotações [Gonçalves and Resende, 2013] (6rBRKGA), e o algoritmo genético híbrido (HGA-S) introduzido em [Feng et al., 2015].

A Tabela 4 compara de forma abrangente as implementações desenvolvidas nas 320 instâncias, considerando a qualidade e a eficiência das soluções. As três primeiras colunas apresentam a classe, as dimensões do contêiner e o número de itens de cada problema. As colunas seguintes mostram o número médio de contêineres (Obj) e o tempo médio de CPU (s) obtidos para as dez instâncias de cada linha da tabela para o algoritmo ACO. As duas últimas colunas mostram o número médio de contêineres (Obj) para os algoritmos 6rBKRGA e HGA-S. Os tempos de CPU não são reportados para esses dois algoritmos. A melhor solução de cada linha é identificada em negrito. Seguindo Mahvash et al. [2017], descartamos as soluções do algoritmo 6rBRKGA para 200 itens nas classes 6, 7 e 8. As soluções apresentadas não respeitam os limites inferiores computados por [Boschetti, 2004] para o 3D-BPP com rotação.

A Tabela 4 revela que o Algoritmo ACO apresenta resultados piores que os algoritmos M-CG e 6rBRKGA, com *gaps* globais de 1,97% e 0,97%, respectivamente, mas supera o algoritmo HGA-S, com um *gap* médio de 0,79%. No entanto, o algoritmo ACO se destaca em termos de eficiência computacional. O ACO não requer mais de 90 segundos para obter soluções com os *gaps* mencionados acima para todos os problemas. O Algoritmo ACO é, em média, cerca de 8,41 vezes mais rápido do que o algoritmo M-CG para obter soluções com o *gap* citado. Esses resultados são significativos se considerarmos que o problema clássico do 3D-BPP precisa ser resolvido diversas vezes como subproblema quando aplicado a problemas reais.

4.2. Grandes Instâncias

Utilizamos os problemas disponíveis para download em http://scm.snu.ac.kr/research/Data_Large.rar para avaliar o comportamento da nossa abordagem em relação às grandes instâncias. As instâncias foram geradas por Thapatsuwan et al. [2012], utilizando um contêiner padrão de vinte pés de unidade equivalente (20 ft x 8 ft x 8 ft). O comprimento, a largura e a altura dos itens seguem distribuições uniformes em [70,100]cm, [50,80]cm e [30,60]cm, respectivamente. São instâncias fortemente heterogêneas no que se refere à diversidade dos itens. Para cada tamanho de problema, foram gerados quinze casos.

Tabela 4: Resultados para instâncias fortemente heterogêneas

Classe	Quantidade de Itens	ACO		M-CG		6rBRKGA	HGA-S
		Obj	CPU(s)	Obj	CPU(s)	Obj	Obj
1	50	11,9	2,6	11,7	10	11,8	11,7
	100	23,7	18,7	22,9	60	23	24,2
	150	33	77,5	31,3	200	31,7	33,5
	200	44,7	58,7	42,8	1000	43,4	47,4
2	50	12,1	2,8	11,8	10	11,8	11,8
	100	23	16	22,4	70	22,5	23,6
	150	33,3	87,7	31,4	100	31,5	32,3
	200	43,9	73,7	41,9	1000	42,5	44,7
3	50	11,9	2,7	11,8	9	11,6	11,9
	100	23,2	16	22,5	60	22,6	23,9
	150	33,8	82,3	32	300	32,4	35,3
	200	44,1	73,1	42,1	1000	42,3	44,4
4	50	28,9	1	28,9	1	28,9	28,8
	100	58,4	1	58,4	3	58,4	58,2
	150	86,4	1	86,4	4	86,4	86,5
	200	118,3	1	118,3	17	118,3	118,3
5	50	7,5	2,7	7,5	5	7,5	7,3
	100	13,9	13,6	13,7	100	13,7	13,9
	150	19,1	84,7	18,6	400	18,6	18,8
	200	25,7	84,6	25,3	1000	25,3	25,9
6	50	8,9	3	8,9	5	9,4	9,1
	100	18	19,5	17,9	100	18,9	18,3
	150	27,9	70,9	27,5	200	28,2	28,2
	200	36	59,8	35,5	1000	33,3	37
7	50	6,5	2,4	6,6	16	6,4	6,3
	100	11,1	13,5	11,2	200	11,3	11,2
	150	14,6	69,6	14,3	500	14,6	15
	200	22,3	77,7	21,9	1000	20,3	22,5
8	50	8,4	2,6	8,3	10	9,2	8,5
	100	17,7	13,1	17,7	22	18,2	18,3
	150	22,9	69,4	22,1	500	22,1	22,5
	200	28,3	74,4	27,7	1000	24,8	29,4
Total All Classes		9194	1177,3	9013	9902	9009	9287

A Tabela 5 apresenta uma comparação do ACO com o algoritmo genético híbrido para grandes instâncias (HGA-L), desenvolvido por Feng et al. [2015]. O HGA-L representa o estado da arte para resolver o 3D-BPP. Para cada tamanho de problema (mostrado na primeira coluna) e método de solução, a Tabela 5 apresenta o espaço desperdiçado médio (AWS) em m^3 e o tempo computacional médio em minutos (CPU). Como os contêineres possuem as mesmas dimensões para todos os problemas, minimizar os espaços ociosos leva aos mesmos resultados que minimizar o número de contêineres. Utilizamos o espaço ocioso como objetivo, conforme Thapatsuwat et al. [2012] e Feng et al. [2015]. Indicamos a melhor solução para cada instância em negrito.

Tabela 5: Resultados para grandes instâncias

Quantidade de Itens	ACO		FENG	
	AWS(m^3)	CPU(min)	AWS(m^3)	CPU(min)
100	12,46	0,21	12,46	1,01
250	48,08	1,58	48,06	2,18
500	89,08	1,15	60,09	6,77
750	108,85	1,02	108,86	9,03
1000	152,78	1,02	118,95	13,46
1250	167,19	1,02	167,19	17,22
1500	186,26	1,02	181,43	20,71
1750	229,31	1,02	212,05	25,90
2000	244,02	1,02	241,61	29,10
2250	286,41	1,02	264,30	35,67
2500	300,73	1,02	300,73	39,14
2750	328,80	1,02	311,89	49,91
3000	355,52	1,02	360,36	52,14
3250	378,14	1,04	370,41	59,88
3500	413,78	1,02	411,31	63,35
3750	431,41	1,03	431,41	69,61
4000	467,46	1,03	472,29	74,94
4250	486,71	1,03	493,95	80,83
4500	515,56	1,03	532,47	86,04
4750	545,22	1,03	550,05	91,45
5000	574,20	1,05	593,53	98,17
Total	6321,96	21,38	6243,40	926,51

Os resultados da Tabela 5 indicam que o algoritmo HGA-L fornece o maior número de melhores soluções (13 instâncias). Esse resultado era esperado, já que esse algoritmo foi especialmente desenvolvido para instâncias grandes, como indicado pelo seu nome. O ACO obtém as melhores soluções em nove casos. Em uma análise global, o ACO aumenta o AWS em cerca de 1,25%, em média, para resolver todas as instâncias. O HGA-L tem melhor desempenho para instâncias de pequeno a médio porte (até 3750 itens), enquanto o ACO apresenta melhores resultados para casos com mais de 4000 itens.

Mas, novamente, é em termos de eficiência que nossos algoritmos se destacam. O ACO reduz o tempo total de CPU necessário para encontrar boas soluções em cerca de 43 vezes. Na maioria dos problemas, o ACO encontra a melhor solução bem antes do tempo indicado na Tabela 5. Esses tempos de CPU envolvem tentativas de melhoria até que o critério de parada no Algoritmo 2 seja atingido. Esse fato explica por que os tempos de CPU do ACO na Tabela 5 são quase independentes da dimensão do problema a partir de 750 itens. Em média, o ACO resolve cada problema em 1,02 minuto, enquanto o HGA-L requer 4,69 e 44,12 minutos, respectivamente.

4.3. Discussão

O algoritmo ACO oferece, em geral, soluções de boa qualidade (encontrando as melhores soluções conhecidas ou com *gaps* razoáveis) de forma muito eficiente. Os melhores resultados foram obtidos para problemas pequenos. Para problemas grandes, o ACO encontrou soluções com um

gap de 1,25% em relação às soluções previamente reportadas na literatura. No entanto, para problemas com mais de 4000 itens, os algoritmos desenvolvidos encontraram as melhores soluções. Ainda assim, os resultados demonstram a versatilidade dos nossos algoritmos na resolução de diferentes classes de problemas. Essa característica foi alcançada ao chamar o algoritmo EP combinando várias regras de rotação e outros critérios de uso do espaço residual, totalizando sete combinações. Esse fato ajuda a otimizar o espaço do contêiner ao gerenciar eficazmente instâncias com poucos e muitos itens, considerando diferentes tamanhos de itens e contêineres.

Além disso, outro atributo importante dos nossos algoritmos desenvolvidos é o bom equilíbrio entre a qualidade e a eficiência do processo de solução, apresentando um desempenho confiável em termos de eficiência da solução. O ACO obtém soluções razoavelmente boas muito rapidamente, reduzindo o tempo necessário para encontrar a solução, em média, para os dois conjuntos de instâncias, em comparação com os resultados previamente reportados. Algumas reduções são significativas, diminuindo o tempo total de CPU dos métodos anteriores entre 9 e 70 vezes. Além disso, uma análise dos resultados obtidos nos dois experimentos indica o potencial de melhorar o desempenho do algoritmo ACO.

O bom compromisso entre qualidade e eficiência e a flexibilidade inédita na resolução de diferentes tamanhos de problemas tornam o algoritmo desenvolvido competitivo como método alternativo para resolver o 3D-BPP com rotação. Sua usabilidade prática torna-se mais relevante ao considerar que o 3D-BPP é um problema clássico que precisa ser customizado por meio da introdução de características reais ou pela sua integração com outros problemas, como roteamento de veículos, para ser útil na prática.

5. Conclusões

Este artigo introduz um algoritmo baseado na otimização de colônia de formigas para resolver o 3D-BPP com rotação. O algoritmo foi extensivamente testado, utilizando problemas de pequeno a grande porte. A abordagem heurística desenvolvida ofereceu resultados competitivos. O ACO encontrou soluções muito rapidamente, mas com alguma perda de eficácia. Recomendamos sua aplicação em problemas de empacotamento *online*, onde o 3D-BPP é resolvido várias vezes como um subproblema, em problemas da classe grande e em circunstâncias onde a qualidade da solução é um critério importante, mas não essencial.

A pesquisa continua com a integração de características reais na abordagem desenvolvida, como a estabilidade dos itens dentro do contêiner e contêineres heterogêneos. A incorporação dessas características aumentará a complexidade do problema, exigindo mudanças em nossos algoritmos para encontrar boas soluções dentro de um tempo aceitável. Além disso, pretendemos incorporar um algoritmo de busca local em nossa abordagem híbrida para melhorar a qualidade da solução.

Referências

- Ali, S., Ramos, A. G., Carravilla, M. A., and Oliveira, J. F. (2022). On-line three-dimensional packing problems: a review of off-line and on-line solution approaches. *Computers & Industrial Engineering*, p. 108122.
- Boschetti, M. A. (2004). New lower bounds for the three-dimensional finite bin packing problem. *Discrete Applied Mathematics*, 140(1-3):241–258.
- Chen, C., Lee, S.-M., and Shen, Q. (1995). An analytical model for the container loading problem. *European Journal of Operational Research*, 80(1):68–76.
- Crainic, T. G., Perboli, G., and Tadei, R. (2008). Extreme point-based heuristics for three-dimensional bin packing. *INFORMS Journal on Computing*, 20(3):368–384.

- Dorigo, M. and Blum, C. (2005). Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344(2):243–278. ISSN 0304-3975. URL <https://www.sciencedirect.com/science/article/pii/S0304397505003798>.
- Feng, X., Moon, I., and Shin, J. (2015). Hybrid genetic algorithms for the three-dimensional multiple container packing problem. *Flexible Services and Manufacturing Journal*, 27:451–477.
- Gonçalves, J. F. and Resende, M. G. (2013). A biased random key genetic algorithm for 2D and 3D bin packing problems. *International Journal of Production Economics*, 145(2):500–510.
- Liu, C., Lyu, J., and Fang, K. (2024). Integrated packing and routing: A model and its solutions. *Computers & Operations Research*, 172:106790.
- Liu, Y., Cao, B., and Li, H. (2021). Improving ant colony optimization algorithm with epsilon greedy and levy flight. *Complex & Intelligent Systems*, 7(4):1711–1722.
- Mahvash, B., Awasthi, A., and Chauhan, S. (2017). A column generation-based heuristic for the three-dimensional bin packing problem with rotation. *Journal of the Operational Research Society*, p. 1–13.
- Martello, S., Pisinger, D., Vigo, D., Boef, E. D., and Korst, J. (2007). Algorithm 864: General and robot-packable variants of the three-dimensional bin packing problem. *ACM Transactions on Mathematical Software*, 33(1):7:1–7:12.
- Pisinger, D. (2002). Heuristics for the container loading problem. *European Journal of Operational Research*, 141(2):382–392.
- Şafak, Ö. and Erdoğan, G. (2023). A large neighbourhood search algorithm for solving container loading problems. *Computers & Operations Research*, 154:106199.
- Thapatsuwan, P., Pongcharoen, P., Hicks, C., and Chainate, W. (2012). Development of a stochastic optimisation tool for solving the multiple container packing problems. *International Journal of Production Economics*, 140(2):737–748.
- Wäscher, G., Haußner, H., and Schumann, H. (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109–1130.