57º SBPO
SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL
Gramado • RS
DE 5 A 9 DE OUTUBRO DE 2025

A Construção de Planos de Empacotamento Retangular Bidimensional com uso do Problema da Mochila

Thiago Martendal Salvador, Rafael de Santiago, Pedro Belin Castellucci
Programa de Pós-Graduação em Ciência da Computação,
Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Cx.P. 476 – 88040-370 Florianópolis – SC – Brasil
thiagomartendal@gmail.com, r.santiago@ufsc.br,
pedro.castellucci@ufsc.br

RESUMO

Este estudo propõe o uso de procedimentos heurísticos para a produção de planos de empacotamento para o Problema de Empacotamento Retangular, visando produzir um empacotamento regular com itens retangulares pequenos em uma folha de estoque única também retangular. Este problema tem grande importância industrial, sendo muito útil para estoque, logística e transportes. E também para a indústria do corte, especialmente cortes de placas ou chapas. Os métodos propostos são inspirados em procedimentos da literatura e constituem heurísticas construtivas capazes de obter taxas de ocupação típicas superiores a 95% para um conjunto de instâncias da literatura.

PALAVRAS CHAVE. Corte e empacotamento, Heurísticas, Problema da mochila.

MH – Metaheurísticas, L&T – Logística e Transportes, OA – Outras Aplicações em PO

ABSTRACT

This study proposes the use of heuristic procedures to produce packing plans for the Rectangular Packing Problem, aiming to produce a regular package with small rectangular items in a single rectangular stock sheet. This problem has great industrial importance, being very useful for storage, logistics and transportation. It is also useful for the cutting industry, especially for cutting plates or sheets. The procedures developed are inspired by works in the literature. The results showed typical occupancy rates above 95% in a dataset from the literature.

KEYWORDS. Cutting and packing, Heuristics, Knapsack problem.

MH – Metaheurístics, L&T – Logistics and Transport, OA – Other Applications in OR

DOI: 10.59254/sbpo-2025-212528
http://live-proceedings.pantheonsite.io/p/212528

1. Introdução

Problemas de empacotamento de retângulos bidimensionais, em folhas também retangulares, são frequentemente estudados dada sua importância prática e teórica. Além de ser possível combinar tais problemas de corte e empacotamento com problemas de dimensionamento de lotes ou roteamento, há aplicações em telecomunicações, logística marítima [Iori et al., 2021] e em indústrias como a madeireira e de vidro [Lodi et al., 2002]. Há diversas variantes para esses problemas, o interesse deste trabalho se concentra nas classes *Single Large Object Placement Problem* (SLOPP) e *Single Knapsack Problem* (SKP), segundo a tipologia de Wäscher et al. [2006], em que se tem uma folha com dimensões conhecidas para acomodar itens com baixa ou alta heterogeneidade, respectivamente.

Devido à NP-dificuldade das variantes SKP e SLOPP, abordagens de enumeração completa possuem aplicação limitada. Em geral, as abordagens exatas encontradas na literatura são algoritmos relativamente sofisticados, como o trabalho seminal de Gilmore e Gomory [1965], que propuseram uma geração de colunas, assim como as propostas de *branch-and-cut-and-price* de Belov e Scheithauer [2006] e de decomposição de *Benders* de Pisinger e Sigurd [2007]. Há também, modelos de Programação Inteira [Beasley, 1985; Furini et al., 2016] e de Programação por Restrições [de Queiroz et al., 2017]. Para uma revisão mais abrangente sobre métodos exatos para o problema pode-se consultar o trabalho de Iori et al. [2021].

Para instâncias maiores, a literatura tem recorrido a heurísticas e metaheurísticas. As metaheurísticas, com frequência, se apoiam em procedimentos de heurísticas construtivas para a solução dos subproblemas envolvidos. É possível encontrar, propostas de heurísticas baseadas em busca tabu [Alvarez-Valdés et al., 2007], *simmulated annealing* [Leung et al., 2012], algoritmo genético [Hadjiconstantinou e Iori, 2007] entre outros. Destaca-se a utilização metaheurística *Pilot Method* por Arruda et al. [2024], que utiliza uma busca em árvore cujo o tamanho é controlado pelo *Pilot Method*. Os resultados apresentados por Arruda et al. [2024] definiram um novo estado-da-arte para diversas instâncias da literatura. Para uma revisão mais abrangente sobre heurísticas pode-se consultar [Lodi et al., 2002; Oliveira et al., 2016].

Entre as propostas de heurísticas construtivas na literatura, destacam-se as de Lodi et al. [1999], Hadjiconstantinou e Iori [2007] e Hifi et al. [2008] por sua similaridade com o proposto neste trabalho. Lodi et al. [1999] consideraram quatro variações do problema de empacotamento em recipientes, em que é necessário minimizar o número de recipientes bidimensionais para posicionar um conjunto de itens disponíveis. A diferença entre as quatro variações se dá devido à possibilidade de rotações ortogonais dos itens e à necessidade de utilizar cortes guilhotinados. Para cada variação foi proposta uma heurística a ser utilizada em um arcabouço único baseado em busca tabu. Uma das heurísticas propostas (FC_{OF}) realiza o empacotamento em níveis, em que a altura de cada nível é definida pela altura do maior item j^* ainda não posicionado. Então, dado o nível definido, se utiliza um algoritmo para o Problema de Mochila 0-1 (KP) para escolher os demais itens que serão posicionados naquele nível. Para o KP, são definidos o peso e o lucro do item respectivamente como sua largura e sua área; e a capacidade da mochila como a largura do recipiente, ou a largura de um espaço vazio sendo preenchido. Na proposta deste trabalho, a altura do nível não é definida antes da solução de KP. Mas o item mais alto dentre os selecionados por KP é que define a altura do nível, aumentando o espaço de busca e, consequentemente, o potencial para soluções de melhor qualidade.

Hadjiconstantinou e Iori [2007] consideraram uma versão do problema de empacotamento de retângulos com um limite de itens disponíveis para cada tipo. Os autores propõem diversos algoritmos gulosos e uma busca baseada em algoritmo genético. Um dos algoritmos gulosos (HC_{KP}) é derivado de Lodi et al. [1999] e funciona essencialmente da mesma forma.

DOI: 10.59254/sbpo-2025-212528
http://live-proceedings.pantheonsite.io/p/212528

Hifi et al. [2008] consideraram uma versão do empacotamento de retângulos similar ao de Hadjiconstantinou e Iori [2007]. Na proposta de solução dos autores a folha (placa) retangular é particionada em níveis utilizando um Problema da Mochila, e são construídas soluções parciais, em cada nível. Então, é utilizado o *Beam Search* para combinar soluções parciais. Nessa proposta, o Problema da Mochila é utilizado de forma similar ao trabalho de Lodi et al. [1999] e a altura de cada nível também é definida *a priori*.

Na proposta deste trabalho, também se utiliza o problema da mochila como subrotina para escolher os itens a serem posicionados nos níveis. Contudo, diferente dos trabalhos encontrados, há uma etapa adicional para preencher os espaços livres de cada nível. O método proposto é uma heurística construtiva simples e eficiente para as variantes SKP e SLOPP do problema de empacotamento de retângulos. Para a solução do problema da mochila foram avaliados um procedimento exato e um heurístico. A heurística é capaz de atingir taxas de ocupação competitivas com métodos mais sofisticados da literatura e mostrou potencial para novas investigações.

O restante do trabalho está organizado da seguinte forma. Na Seção 2, é definido formalmente o problema de interesse e apresentado o método de solução. Na sequência, os experimentos realizados são descritos na Seção 3. Por fim, na Seção 4, são tecidas conclusões e feitas menções a trabalhos futuros.

2. Definição do Problema e Método de Solução Proposto

Nesta seção são apresentados a definição formal do problema de interesse, Subseção 2.1, e o método de solução proposto, Subseção 2.2, juntamente com um exemplo e uma descrição das subrotinas para o Problema da Mochila. A definição é compatível com outros trabalhos da literatura para as versões bidimensionais de problemas das classes SLOPP e SKP, e o método proposto realiza o preenchimento da folha de estoque em níveis.

2.1. Definição do Problema

Seja uma folha de estoque F definida por uma dupla $F = (W, H)$, na qual $W \in \mathbb{Z}^+$ representa a largura da folha e $H \in \mathbb{Z}^+$ representa a altura da folha. Considere também um conjunto de itens $I = \{1, 2, \ldots, n\}$, no qual cada item possui uma largura dada pela função $w : I \to \mathbb{Z}^+$ e uma altura $h : I \to \mathbb{Z}^+$. Deseja-se encontrar o subconjunto $S \subseteq I$, tal que o valor

$$H \times W - \sum_{i \in S} w(i)h(i)$$

seja mínimo, e que todo item $i \in S$ deve estar empacotado F, em coordenada x_i e y_i, $0 \leq x_i \leq W - w(i)$, $0 \leq y_i \leq H - h(i)$, Além disso, nenhum par de itens $(i, j) \in S \times S$ não podem estar sobrepostos, ou seja, se $y_i \in [y_j, y_j + h_j)$, então $x_i + w_i < x_j$, ou se $y_j \in [y_i, y_i + h_i)$, então $x_j + w_j < x_i$. Por fim, os itens devem ser posicionados com suas faces paralelas às da folha de estoque.

2.2. Método Proposto

O método proposto iterativamente seleciona uma área retangular vazia da folha (subespaço da folha completa) e a preenche com itens. Em cada retângulo vazio, empacotam-se itens um ao lado do outro em uma área retangular vazia. O procedimento pára quando não existir área retangular vazia que possa acomodar um item na folha. O procedimento é apresentado no Algoritmo 1.

Das linhas 1 à 8, prepara-se as estruturas de dados que serão utilizadas durante o processo. No conjunto C, estarão itens que serão empacotados pelo método. W' e H' correspondem à largura e altura, respectivamente, do retângulo vazio a ser preenchido, que no primeiro momento é a própria folha. I' é o conjunto de itens que ainda não foram empacotados. A princípio, o conjunto é composto por todos os itens. As variáveis x e y correspondem a coordenada do canto inferior esquerdo

DOI: 10.59254/sbpo-2025-212528
http://live-proceedings.pantheonsite.io/p/212528

do retângulo vazio a ser preenchido. Os vetores X e Y possuem em cada posição i a coordenada x e y do item i, respectivamente. O valor **null** marca a posição de um item não empacotado.

Antes de cada iteração do laço de repetição da linha 10, um subconjunto I^R é formado, contendo os itens que caibam (individualmente) no retângulo vazio e que ainda não foram empacotados (linhas 9 e 16). Eles são considerados elegíveis ao empacotamento no retângulo vazio, na próxima iteração. A cada iteração, utiliza-se um método de solução para o problema da Mochila para decidir quais itens do conjunto I^R serão empacotados (linha 11). Ele seleciona os itens mais valiosos a serem empacotados no retângulo vazio, com largura e altura estabelecidos (respectivamente W' e H'). É importante informar que apenas os itens que tenham a altura menor ou igual a do retângulo vazio são considerados. Os itens selecionados são gravados no conjunto I'' e logo em seguida são empacotados em fila (um ao lado do outro, através do método *empacotarItens*, na linha 12), de modo que não ultrapassem a largura e a altura do retângulo vazio utilizado (algo garantido pelo conjunto de itens elegíveis em I^R). Neste processo de empacotamento, as posições dos itens empacotados são gravadas nas estruturas X e Y. Logo depois, atualiza-se o conjunto de itens que ainda não foram empacotados I' e o conjunto de itens empacotados C (linhas 13 e 14). Na linha 15, o próximo retângulo vazio é localizado (procedimento *proximoEspacoLivre*), considerando que a busca siga o espaço vazio imediatamente à direita e acima de itens já empacotados ou do limite inferior da folha, no qual caiba pelo menos um item do conjunto I'. Os dados referentes ao retângulo vazio são gravados em W', H', x, y, que são comprimento, largura e a coordenada x e y do canto inferior do retângulo, respectivamente. Caso nenhum retângulo vazio seja localizado, a largura e a altura retornadas pelo procedimento serão 0. Como nenhum item caberá neste espaço, o I^R estará vazio e o laço da linha 10 pára. O laço também se encerra caso não haja mais itens a serem empacotados. Nas linhas 18 e 19, o valor da solução e a área ocupada são calculados e retornados pelo procedimento proposto, juntamente com as coordenadas e o conjunto de itens empacotados.

Exemplo de Funcionamento do Método de Solução

Para ilustrar o funcionamento do método de solução proposto, a seguir é apresentado um exemplo de empacotamento. Para isso, é importante ressaltar que quando uma seleção de itens é realizada, os itens selecionados são empacotados sequencialmente, um a um, na ordem em que foram selecionados por um método de solução para o problema da mochila. A Figura 1 exemplifica os passos iniciais de construção da solução. A folha de estoque é apresentada na Figura 1A, estando inicialmente vazia para receber os itens, considerada como primeiro retângulo vazio. A Figura 1B apresenta os primeiros itens alocados que foram obtidos com uma chamada do método de solução da mochila, com os espaços vazios demarcados em vermelho, entre os itens e a demarcação em azul do item mais alto. Como os itens são inseridos na ordem em que foram selecionados, o item mais alto pode ser alocado em qualquer lugar, e não apenas em um dos cantos. Os espaços vazios são numerados de acordo com sua ordem de formação e inseridos em uma lista de espaços vazios, que mantém a ordem em que os espaços são inseridos. Note também que os espaços vazios são limitados acima pela altura da faixa atual.

A Figura 1C apresenta o empacotamento dos espaços livres. Quando um espaço livre é identificado, este é selecionado. O empacotamento é feito exatamente como na primeira etapa, sendo também preparada uma instância do problema da mochila para selecionar itens que possam caber nas dimensões do retângulo vazio em questão. Junto o empacotamento em um retângulo vazio.

A Figura 1D apresenta um conjunto de itens completamente empacotado em seu retângulo, anteriormente vazio, e logo acima aparecem em vermelho os próximos retângulos vazios a serem preenchidos por novas iterações do procedimento proposto.

DOI: 10.59254/sbpo-2025-212528
http://live-proceedings.pantheonsite.io/p/212528

Algoritmo 1: Empacotamento Proposto

Entrada: a largura da folha $W \in \mathbb{Z}^+$, a altura da folha $H \in \mathbb{Z}^+$, o conjunto de itens a empacotar I, as funções de largura, altura e avaliação de cada item $w : I \to Z^+$, $h : I \to Z^+$ e $v : I \to Z^+$ (respectivamente).

Saída: a avaliação obtida na solução proposta, a área ocupada, o posicionamento de cada item e o conjunto de itens empacotados.

// Inicialização
$C \leftarrow \{\}$ // conjunto de itens empacotados
$W' \leftarrow W$ // largura do retângulo vazio
$H' \leftarrow H$ // altura do retângulo vazio
$I' \leftarrow I$ // itens não empacotados
$x \leftarrow 0$ // posição de referência coordenada x
$y \leftarrow 0$ // posição de referência coordenada y
$X[i] \leftarrow$ **null** $\forall i \in I'$ // coordenada x do item
$Y[i] \leftarrow$ **null** $\forall i \in I'$ // coordenada y do item
// Processo iterativo de empacotamento
$I^R \leftarrow \{i \in I' : h(i) \leq H' \wedge w(i) \leq W'\}$ // Itens para o retângulo vazio
enquanto $I^R \neq \{\}$ E $I' \neq \emptyset$ **faça**
  // Seleciona os itens
  $I'' \leftarrow$ mochila(I^R, w, v, W')
  empacotarItens(I'', x, y, X, Y)
  $I' \leftarrow I' - I''$
  $C \leftarrow C \cup I''$
  // Busca próximo retângulo vazio
  $W', H', x, y \leftarrow$ proximoEspacoLivre(x, y, X, Y, I')
  // Itens para o retângulo vazio
  $I^R \leftarrow \{i \in I' : h(i) \leq H' \wedge w(i) \leq W'\}$
fim
$valor \leftarrow \sum_{i \in C} v(i)$
$area \leftarrow \sum_{i \in C} w(i) \cdot h(i)$
retorna $valor, area, X, Y, C$

Métodos de Solução para o Problema da Mochila

No contexto deste trabalho, a mochila é o retângulo vazio selecionado iterativamente pelo método proposto. A função de avaliação utilizada é a área de cada item, ou seja, para cada item i, $v(i) = w(i)h(i)$. Os itens recebidos são subconjunto de itens que cabem na altura e na largura do retângulo vazio. Então, o método da mochila usa a largura do retângulo como capacidade da mochila (W) para colocar os itens horizontalmente alinhados lado-a-lado.

Dois métodos de solução foram utilizados para a mochila resolver o problema de alocar itens em cada retângulo vazio: para mochila binária (Algoritmo 2) e para mochila fracionária (Algoritmo 3).

O método de solução para mochila binária (Algoritmo 2) utiliza-se do algoritmo de programação dinâmica. Nele, a matriz de memorização $M[n', w']$ é preenchida com a solução ótima para o subproblema considerando os itens $1, 2, \ldots, n'$ e uma mochila com capacidade n'. O preenchimento desta matriz ocorre entre as linhas 1 e 12. Depois, a solução ótima para o problema estará na posição $M[n, W]$. Então, os itens são mapeados por identificar as subsoluções

DOI: 10.59254/sbpo-2025-212528
http://live-proceedings.pantheonsite.io/p/212528

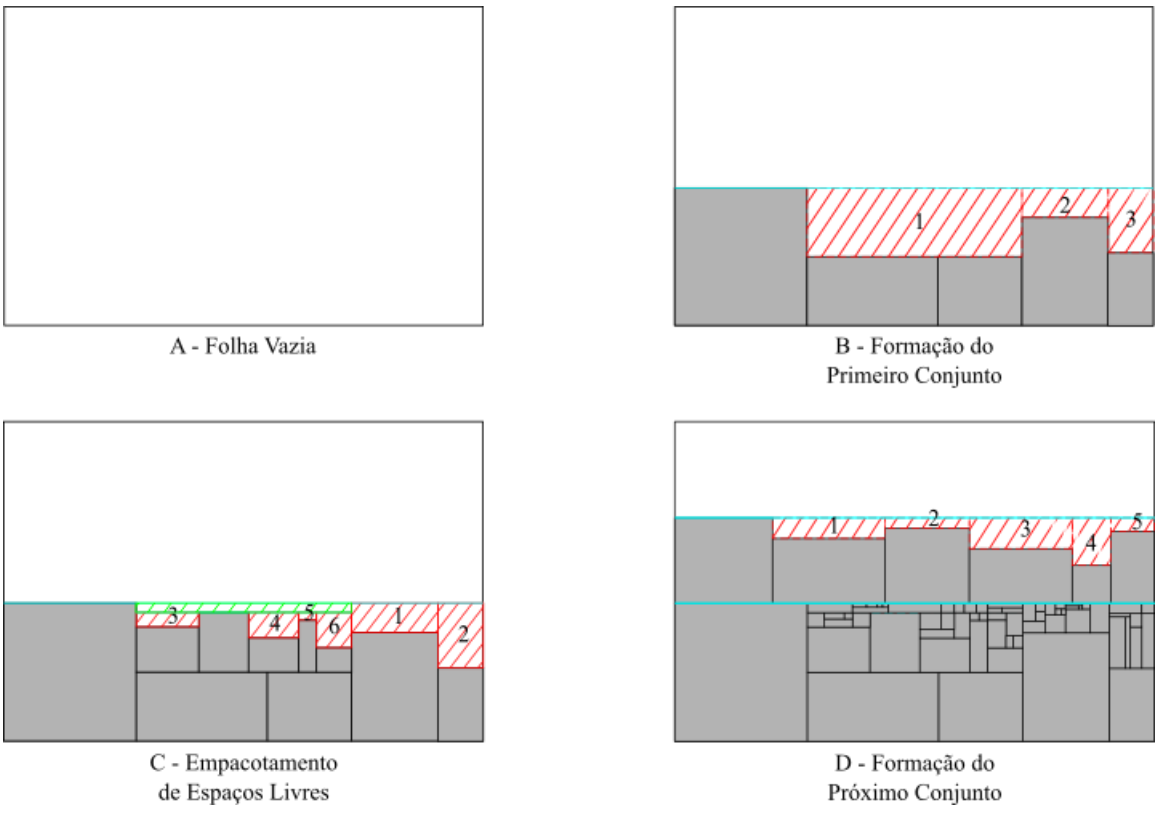


Figura 1: As etapas de solução.

ótimas utilizadas para definir a solução ótima para o problema varrendo a matriz M (linhas 13 à 25). Finalmente o conjunto de itens é retornado.

Para o método de solução da mochila fracionária (Algoritmo 3), utiliza-se uma busca gulosa cujo o critério é selecionar prioritariamente os itens i que tenham o maior $\frac{v(i)}{w(i)}$. Esta avaliação é gravada pelo método na função v' (linhas 1 à 4). Esta função é utilizada para ordenar os itens em ordem decrescente pelo valor nela estabelecido para cada item (linha 5). Depois, iterativamente (linhas 8 à 13) selecionam-se os itens de maior prioridade (valor na função v') e é verificado se cabem na mochila. Caso seja verdadeiro, a capacidade ocupada e o conjunto de itens selecionados são atualizados. Ao final, retorna-se o conjunto de itens inseridos na mochila.

3. Experimentos

O procedimento apresentado anteriormente foi implementado em linguagem C++ sendo compilado em um compilador GNU G++ na versão 13.3.0. A partir da implementação realizada, foram feitos experimentos para se determinar a qualidade das soluções. A realização dos experimentos consiste na execução do procedimento sob instâncias de dados de entrada *zdf*, obtidas do site da ESICUP – *EURO Special Interest Group on Cutting and Packing* [ESICUP, 2024]. Estas instâncias se tratam de arquivos de texto que contém os itens para avaliação, onde cada item possui um índice e suas dimensões de largura e altura. A quantidade de itens em cada instância foi disponibilizada na Tabela 1. A execução dos experimentos foi feita em um computador com processador Core i7-8550U CPU em 1.80GHz e 8 Gb de memória RAM.

A Tabela 1 foi obtida de Arruda et al. [2024] e apresenta resultados obtidos de heurísticas da literatura (HSA [Leung et al., 2012] e IBHP [Shiangjen et al., 2017]) e das heurísticas propostas no trabalho (IN-H0, IN-H1 e IN-H2). Os valores apresentados em cada heurística da Tabela 1 são

DOI: 10.59254/sbpo-2025-212528
http://live-proceedings.pantheonsite.io/p/212528

Algoritmo 2: Algoritmo para o Problema da Mochila Binária.

Entrada: um conjunto de itens $I = \{1, 2, \ldots, n\}$, uma função de pesos $w : I \to \mathbb{Z}^+$, uma função de valor $v : I \to \mathbb{R}$ e uma capacidade limite $W \in \mathbb{Z}^+$

Saída: Itens escolhidos na solução do problema da mochila.

```
// Matriz de memorização
```

Criar matriz $M \in \mathbb{R}^{(|I|+1)\times(W+1)}$
$M[0, w] = 0 \; \forall w \in \{0, 1, 2, \ldots, W\}$
para cada $i = 1, 2, \ldots, n$ **faça**
  para $w \leftarrow 0$ ***to*** W **faça**
    se $w < w(i)$ **então**
      $M[i, w] \leftarrow M[i-1, w]$
    fim
    senão
      $M[i, w] \leftarrow \mathbf{max}\{M[i-1, w], v(i) + M[i-1, w-w(i)]\}$
    fim
  fim
fim

```
// identificando os itens adicionados na solução ótima
```

itens $\leftarrow \{\}$
$i \leftarrow n$
$j \leftarrow W$
enquanto $i \geq 1$ **faça**
  enquanto $i \geq 1$ ***e*** $M[i, j] = M[i-1, j]$ **faça**
    $i \leftarrow i - 1$
  fim
  se $i \geq 1$ **então**
    itens $\leftarrow$ itens $\cup\{i\}$
    $j \leftarrow j - w(i-1)$
    $i \leftarrow i - 1$
  fim
fim
retorna $itens$

DOI: 10.59254/sbpo-2025-212528
http://live-proceedings.pantheonsite.io/p/212528

Algoritmo 3: Método de Solução para o Problema da Mochila Fracionária.

Entrada: um conjunto de itens $I = \{1, 2, \ldots, n\}$, uma função de pesos $w : I \to \mathbb{Z}^+$, uma função de valor $v : I \to \mathbb{R}$ e uma capacidade limite $W \in \mathbb{Z}^+$
Saída: Itens escolhidos na solução do problema da mochila.

```
// Critério de custo
criar nova função v' : I → ℝ+
para cada i ∈ I faça
    definir v'(i) = v(i)/w(i)
fim
// ordena os itens na ordem decrescente dada por v'
I' ← ordenarDecrescente(I, v')
capacidade ← 0
itens ← {}
para cada i ∈ I' faça
    se capacidade+w(i) ≤ W então
        capacidade ← capacidade +w(i)
        itens ← itens ∪{i}
    fim
fim
retorna itens
```

Tabela 1: Resultados da literatura com tempo limite de 60s. Tabela adaptada de [Arruda et al., 2024].

Instância	Itens na Instância	Dimensão da Folha	HSA	IBHP	IN-H0	IN-H1	IN-H2
zdf1	580	100x330	36	36	36	36	36
zdf2	660	100x357	20	20	20	20	20
zdf3	740	100x384	40	40	40	40	40
zdf4	820	100x407	8	8	8	8	8
zdf5	900	100x434	80	80	80	80	80
zdf6	1532	3000x4872	1033710	748192	776229	776229	967172
zdf7	2432	3000x4852	1081572	672592	776229	740192	776229
zdf8	2532	3000x5172	852772	268683	366237	268683	381742
zdf9	5032	3000x5172	836260	5976	339116	268683	182492
zdf10	5064	6000x5172	811780	14792	1813	287184	287184
zdf11	7564	6000x5172	742108	6260	543463	6347	9372
zdf12	10064	6000x5172	537436	2236	583702	5147	5147
zdf13	15096	9000x5172	270448	3592	269667	11566	11566
zdf14	25032	3000x5172	735467	492	268683	492	492
zdf15	50032	3000x5172	366237	492	492	492	492
zdf16	75032	3000x5172	492	492	492	492	492

DOI: 10.59254/sbpo-2025-212528
http://live-proceedings.pantheonsite.io/p/212528

Tabela 2: Execução do procedimento utilizando a mochila 0-1.

Instância	Dimensão da Folha	Itens Alocados	Área Ocupada	Perda de Corte	Taxa de Ocupação (%)	Tempo Total (ms)
zdf1	100x330	515	31968	1032	96.88	161
zdf2	100x357	631	34906	794	97.78	193
zdf3	100x384	714	37778	622	98.39	221
zdf4	100x407	802	40390	310	99.24	278
zdf5	100x434	885	43076	324	99.26	316
zdf6	3000x4872	1530	13475690	1140310	92.20	1452
zdf7	3000x4852	2429	12948906	1607094	88.96	3528
zdf8	3000x5172	2527	15135233	380767	97.55	3331
zdf9	3000x5172	5027	15135233	380767	97.55	13011
zdf10	6000x5172	5055	30394518	637482	97.95	32820
zdf11	6000x5172	7555	30394518	637482	97.95	66469
zdf12	6000x5172	10060	30665592	366408	98.82	86331
zdf13	9000x5172	15084	45761427	786573	98.32	214608
zdf14	3000x5172	25027	15135233	380767	97.55	138178
zdf15	3000x5172	50027	15135233	380767	97.55	413081

a perda de corte (área total da folha menos a área dos itens empacotados) obtida com o empacotamento realizado, onde os valores sublinhados indicam que todos os itens disponíveis na instância foram empacotados, não foram disponibilizadas as quantidades de itens empacotadas para os demais casos. As dimensões de folha de estoque utilizadas aqui são consideradas para os experimentos, para se ter um referencial comparativo.

Tabela 3: Execução do procedimento utilizando a mochila fracionária.

Instância	Dimensão da Folha	Itens Alocados	Área Ocupada	Perda de Corte	Taxa de Ocupação (%)	Tempo Total (ms)
zdf1	100x330	512	32085	915	97.23	14
zdf2	100x357	579	34483	1217	96.60	19
zdf3	100x384	689	37625	775	97.99	24
zdf4	100x407	784	40119	581	98.58	27
zdf5	100x434	871	42821	579	98.67	37
zdf6	3000x4872	1530	13296090	1319910	90.97	67
zdf7	3000x4852	2430	13236090	1319910	90.94	140
zdf8	3000x5172	2530	14981199	534801	96.56	94
zdf9	3000x5172	5030	14981199	534801	96.56	239
zdf10	6000x5172	5059	30277982	754018	97.58	512
zdf11	6000x5172	7554	30232066	799934	97.43	987
zdf12	6000x5172	10059	30277982	754018	97.58	1385
zdf13	9000x5172	15084	45599699	948301	97.97	2934
zdf14	3000x5172	25030	14981199	534801	96.56	3608
zdf15	3000x5172	50030	14981199	534801	96.56	15642
zdf16	3000x5172	75030	14981199	534801	96.56	39651

DOI: 10.59254/sbpo-2025-212528
http://live-proceedings.pantheonsite.io/p/212528

As Tabelas 2 e 3 apresentam os resultados de execução dos experimentos. Nestas tabelas, as colunas após **Dimensão da Folha** apresentam os resultados para mensurar a qualidade das soluções, mas apenas os dados da coluna de **Perda de Corte** serão utilizados para a comparação com a Tabela 1. Então, comparando as perdas de corte registradas para cada heurística da Tabela 1 com as perdas de corte apresentadas nas Tabelas 2 e 3, observa-se que a heurística proposta neste estudo não conseguiu superar nenhuma das heurísticas da literatura consideradas na Tabela 1. Porém, os resultados obtidos nos experimentos conseguiram apresentar altas taxas de ocupação em um tempo de execução muito viável, especialmente os resultados obtidos utilizando a mochila fracionária (na Tabela 3) que tiveram todos os tempos de execução inferiores a um minuto. É importante notar que na Tabela 2 não foram apresentados os resultados para a instância *zdf16*, pois a execução com o procedimento de mochila 0-1 não conseguiu ser finalizada para uma folha de dimensão 3000x5172, devido o consumo de memória que as chamadas da mochila 0-1 demandaram para os 75032 itens da instância. Isso fez com que a execução fosse interrompida.

É importante observar que em alguns casos o procedimento utilizando a mochila fracionária conseguiu atingir uma maior taxa de ocupação, ao mesmo tempo que a perda de corte foi menor. Isso ocorre em casos como as instâncias *zdf1* e *zdf7*, mas de forma geral, o uso da mochila 0-1 como auxiliar proporcionou maiores taxas de ocupação. No entanto, os resultados obtidos com a mochila fracionária mostram que esta mochila é uma alternativa muito interessante por ter proporcionado tempos de execução muito pequenos e por conseguir lidar com instâncias com mais de 50000 itens em uma folha de dimensão mais elevada, já que foi obtido um resultado para a instância *zdf16*.

No geral, apesar das taxas de ocupação inferiores ao observado no estado-da-arte da literatura, os resultados são promissores, com diferenças sempre inferiores a 5% das soluções obtidas por algoritmos do estado-da-arte. Principalmente no caso da utilização da mochila fracionária, os tempos computacionais não ultrapassaram 40 segundos inclusive nas maiores instâncias. Além disso, o método proposto é uma heurística construtiva, consideravelmente mais simples do que a busca em árvore, recursiva e com mecanismos de poda, proposta por Arruda et al. [2024].

4. Conclusão

As versões bidimensionais do problema de empacotamento de retângulos SLOPP e SKP possuem diversas aplicações práticas e propostas de métodos de solução, tanto exatos quanto heurísticos. Diversos métodos heurísticos, como o proposto neste trabalho, utilizam-se de versões unidimensionais de problemas de corte e empacotamento como subrotina para construção de uma solução.

A abordagem proposta, utiliza-se da ideia de construir uma solução em níveis (ou prateleiras). Contudo, adiciona uma etapa de melhoramento da área ocupada em cada nível. Tanto os itens para construção do nível quanto para aumentar sua taxa de ocupação são escolhidos com base na solução de instâncias de problemas da mochila 0-1 (KP).

Foram avaliadas duas opções de procedimentos para a solução do KP, uma exata, baseada em Programação Dinâmica, e uma heurística, com base em um procedimento guloso. A versão com solução exata, tipicamente, resulta em uma solução com menor desperdício de espaço. Embora implique em um maior uso de recurso computacional, tanto memória quanto tempo.

As taxas de ocupação típicas foram superiores a 95% para um conjunto de instâncias da literatura. Portanto, pode ser interessante utilizar o procedimento proposto combinado com heurísticas de melhoria e metaheurísticas. Além disso, a experimentação com outros conjuntos de instâncias beneficiaria a caracterização do comportamento do método proposto. Por fim, outros trabalhos futuros poderiam investigar a adaptação do método para outras variantes como o empacotamento em faixas (*strip packing*) e a possibilidade de rotação dos itens. Além de alternativas para

DOI: 10.59254/sbpo-2025-212528
http://live-proceedings.pantheonsite.io/p/212528

a solução do problema de mochila.

Referências

Alvarez-Valdés, R., Parreño, F., e Tamarit, J. M. (2007). A tabu search algorithm for a two-dimensional non-guillotine cutting problem. *European Journal of Operational Research*, 183 (3):1167–1182.

Arruda, V. P. d. R., Mirisola, L. G. B., e Soma, N. Y. (2024). Rectangle packing with a recursive pilot method. *Computers & Operations Research*, 161:106447. ISSN 0305-0548.

Beasley, J. E. (1985). An exact two-dimensional non-guillotine cutting tree search procedure. *Operations research*, 33(1):49–64.

Belov, G. e Scheithauer, G. (2006). A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting. *European journal of operational research*, 171(1):85–106.

de Queiroz, T. A., Hokama, P. H. D. B., Schouery, R. C. S., e Miyazawa, F. K. (2017). Two-dimensional disjunctively constrained knapsack problem: Heuristic and exact approaches. *Computers & Industrial Engineering*, 105:313–328.

ESICUP (2024). Data sets - esicup - euro special interest group on cutting and packing. Acessado em 21 de agosto de 2024.

Furini, F., Malaguti, E., e Thomopulos, D. (2016). Modeling two-dimensional guillotine cutting problems via integer programming. *INFORMS Journal on Computing*, 28(4):736–751.

Gilmore, P. C. e Gomory, R. E. (1965). Multistage cutting stock problems of two and more dimensions. *Operations research*, 13(1):94–120.

Hadjiconstantinou, E. e Iori, M. (2007). A hybrid genetic algorithm for the two-dimensional single large object placement problem. *European Journal of Operational Research*, 183(3):1150–1166.

Hifi, M., M'Hallah, R., e Saadi, T. (2008). Algorithms for the constrained two-staged two-dimensional cutting problem. *INFORMS Journal on Computing*, 20(2):212–221.

Iori, M., De Lima, V. L., Martello, S., Miyazawa, F. K., e Monaci, M. (2021). Exact solution techniques for two-dimensional cutting and packing. *European Journal of Operational Research*, 289(2):399–415.

Leung, S. C., Zhang, D., Zhou, C., e Wu, T. (2012). A hybrid simulated annealing metaheuristic algorithm for the two-dimensional knapsack packing problem. *Computers & Operations Research*, 39(1):64–73. ISSN 0305-0548. Special Issue on Knapsack Problems and Applications.

Lodi, A., Martello, S., e Monaci, M. (2002). Two-dimensional packing problems: A survey. *European journal of operational research*, 141(2):241–252.

Lodi, A., Martello, S., e Vigo, D. (1999). Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. *INFORMS journal on computing*, 11(4):345–357.

Oliveira, J. F., Neuenfeldt, A., Silva, E., e Carravilla, M. A. (2016). A survey on heuristics for the two-dimensional rectangular strip packing problem. *Pesquisa Operacional*, 36(2):197–226.

DOI: 10.59254/sbpo-2025-212528
http://live-proceedings.pantheonsite.io/p/212528

Pisinger, D. e Sigurd, M. (2007). Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem. *INFORMS Journal on Computing*, 19(1): 36–51.

Shiangjen, K., Chaijaruwanich, J., Srisujjalertwaja, W., Unachak, P., e Somhom, S. (2017). An iterative bidirectional heuristic placement algorithm for solving the two-dimensional knapsack packing problem. *Miscellaneous*, 50(2):347–365.

Wäscher, G., Haußner, H., e Schumann, H. (2006). An improved typology of cutting and packing problems. *Elsevier ScienceDirect Journals*, 183(3):1109–1130.

DOI: 10.59254/sbpo-2025-212528
http://live-proceedings.pantheonsite.io/p/212528