LI Simpósio Brasileiro de Pesquisa Operacional
Limeira–SP, 02 a 06 de setembro de 2019

Uma heurística baseada em programação lógica por restrições para um problema restrito de corte bidimensional guilhotinado

Junot Freire, Michell Queiroz, Luis Modesto, Mateus Carvalho, Rafael Melo
Departamento de Ciência da Computação, Universidade Federal da Bahia
Avenida Adhemar de Barros, s/n, Salvador, BA 40170-110, Brasil
{junotsantos,melo, michellfelippe}@dcc.ufba.br,
mateuscsilva.1@gmail.com, luis-modesto@outlook.com

RESUMO

Problemas de corte bidimensional (2BP, do inglês *2-Dimensional bin packing problem*) são problemas clássicos de otimização combinatória e da indústria têxtil, metalúrgica e de vidro, e consistem em alocar um conjunto de itens retangulares em placas retangulares maiores com tamanho padronizado com a finalidade de minimizar o desperdício de matéria-prima. No presente trabalho apresenta-se uma abordagem para um problema restrito de corte bidimensional guilhotinado considerado no *ROADEF/EURO Challenge 2018: Cutting Optimization Problem*, no qual existe a possibilidade de rotacionar itens em 90° e as placas retangulares a serem cortadas podem possuir defeitos em determinados pontos. Propõe-se uma heurística *multi-start* gulosa randomizada e técnica de aprimoramento que combina a heurística com uma modelagem de programação lógica por restrições. Experimentos realizados mostram que a abordagem proposta é capaz de produzir resultados competitivos que se aproximam dos melhores conhecidos, tendo classificado a equipe para a etapa final do *ROADEF/EURO Challenge 2018*.

PALAVRAS CHAVE. Problema de corte bidimensional, Heurísticas, Programação lógica por restrições

ABSTRACT

2-Dimensional bin packing problems (2BP) are classic combinatorial optimization problems and in textile, metallurgical and glass industries. 2BP consists in allocating a set of rectangle items in bigger rectangle standardized plates aiming to minimize raw material waste. In this work, we approach a restrict version of the 2BP with guillotine cuts presented in the *ROADEF/EURO Challenge 2018: Cutting Optimization Problem*, in which there is a possibility to rotate items in 90° and the rectangle plates may present defects in certain points. We propose a multi-start greedy randomized heuristic and an improving technique that combines the heuristic with a constraint programming model. Computational experiments show that the proposed technique is capable of generate competitive results with the best ones avaiable, which qualified the team for the finals in the *ROADEF/EURO Challenge 2018*.

KEYWORDS. 2-Dimensional Bin Packing Problem, Heuristics, Constraint programming

https://proceedings.science/p/106878?lang=pt-br

1. Introdução

Em diversas atividades industriais, tais como na metalurgia e indústria de vidros, é preciso produzir (ou empacotar) um conjunto de itens retangulares a partir de grandes placas retangulares de tamanho padronizado, enquanto minimiza-se os desperdícios de matéria-prima decorrentes destas aplicações. Este tipo de atividade está relacionada a um problema clássico de otimização combinatória, denominado Problema de Corte Bidimensional (2BP, do inglês *2-Dimensional Bin Packing Problem*).

Devido ao grande interesse comercial atribuído a este problema, o 2BP é recorrente na literatura (Velasco e Barboza [2014]) e possui uma grande quantidade de variantes para atender diferentes demandas da indústria. O 2BP consiste em, dado um conjunto de placas retangulares, com largura e altura fixadas, e um conjunto de itens retangulares, onde cada item possui valores de largura e altura específicos, é preciso dispor, sem sobreposição, todos os itens no menor conjunto de placas possível. Os itens serão produzidos a partir de um processo de corte realizado na matéria-prima fornecida, i.e., as placas. O 2BP é NP-Difícil, visto que é um caso mais geral do problema de corte unidimensional, que é conhecido como sendo NP-Difícil (Lodi et al. [2002]).

2BP possui uma vasta gama de variações, dentre as quais pode-se citar a rotação de itens no processo de corte, os tipos e limitações dos cortes, e a presença de defeitos e avarias na matéria-prima a ser utilizada. Em diversos trabalhos sobre corte bidimensional na literatura, os itens a serem produzidos na placa podem ou não serem rotacionados em um ângulo de 90°. Não se costuma permitir a rotação das placas de matéria-prima, devido ao fato das placas possuírem dimensões muito grandes, quando comparadas aos itens, e sua rotação na cadeia de produção ser uma tarefa pouco prática. Wuttke e Heese [2018] apresentam uma heurística e uma formulação de programação inteira mista para uma variação do 2BP onde a rotação de itens não é permitida e deseja-se minimizar o tempo de produção dos itens. Já Furini et al. [2012] propõem uma heurística baseada em um modelo de geração de colunas para o 2BP onde a rotação de itens é permitida, com o objetivo de minimizar a área total de placas utilizadas.

Um dos tipos mais comuns de variação do 2BP é quanto ao tipo de corte, que é classificado como: (a) guilhotinado, no qual o corte começa de um lado da placa e termina no lado paralelo oposto, e produz como resultado duas placas retangulares, e (b) não-guilhotinado, no qual o corte é realizado de acordo com as dimensões do item, não sendo necessário cortar a placa de uma extremidade a outra. Em Dolatabadi et al. [2012] são apresentados algoritmos exatos para resolução do problema de corte bidimensional guilhotinado sem rotação de itens. Alvarez-Valdés et al. [2007] apresentam uma abordagem de busca tabu para resolver o problema de corte bidimensional sem rotação de itens e com corte não guilhotinado.

Algumas linhas de produção de produtos que utilizam cortes guilhotinados também possuem limitações sobre quantas vezes é possível alternar entre cortes verticais e horizontais. Um corte guilhotinado é denominado *k-cut* quando ele alterna k vezes entre cortes verticais e horizontais. Dentre os casos mais comuns destas limitações estão o *2-cut* e *3-cut*, ou também denominados de corte em 2 e 3 estágios, como visto em Silva et al. [2010].

Além das variações apresentadas anteriormente, nem todo problema de corte guilhotinado possui placas em perfeitas condições, onde trincos e rachaduras podem ocorrer ao final do processo de fabricação das placas. A presença desses defeitos dificulta o processo de corte, como visto em Beasley [2004]. Caso a ferramenta de corte atravesse um defeito (como um trinco ou rachadura) é possível que a área defeituosa se expanda ou que a placa seja quebrada. Portanto, nenhum item pode conter avarias e nenhum corte deve ser realizado por cima de um defeito. A Figura 1 apresenta um exemplo de plano de corte guilhotinado com existência de três defeitos (retângulos mais escuros) e

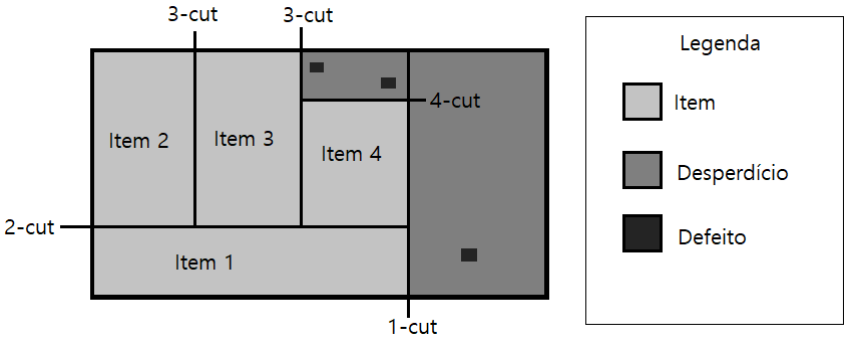
https://proceedings.science/p/106878?lang=pt-br

cortes indo até o nível *4-cut*, para separar um item de um defeito.

Neste trabalho propõem-se uma heurística gulosa randomizada *multi-start* e uma técnica de melhoria baseada em um modelo de programação lógica por restrições (do inglês, *constraint programming*) para um caso particular do problema de corte bidimensional com corte guilhotinado e rotação de itens. O problema foi apresentado no *ROADEF/EURO Challenge 2018: Cutting Optimization Problem*[1], que demonstra motivação proveniente das linhas de produção de algumas indústrias de corte de vidro. As técnicas propostas garantiram a classificação deste trabalho para a fase final do *ROADEF/EURO Challenge 2018*.

O restante do trabalho é organizado da seguinte forma. A Seção 2 apresenta uma descrição formal do problema. A Seção 3 descreve a heurística gulosa randomizada *multi-start* e a técnica de aprimoramento baseada em programação lógica por restrições. A Seção 4 resume os experimentos computacionais para instâncias *benchmark* do *ROADEF/EURO Challenge 2018*. Comentários finais e propostas de trabalhos futuros são discutidos na Seção 5.

Figura 1: Exemplo de corte guilhotinado com defeitos e 4-cut.



2. O problema

Linhas de produção de indústrias de corte de vidro fazem uso de uma técnica de fabricação denominada *float process*, onde o vidro líquido recebe um banho de estanho, é resfriado e depois cortado em grandes placas chamadas *jumbos*. Esses *jumbos* são empilhados e enviados para o processo de corte, a fim de gerar itens menores de acordo com as demandas dos clientes.

Os *jumbos* podem apresentar pequenos defeitos decorrentes do *float process*, como pequenos trincos. Ao final do processo de produção das placas, um sensor analisa os *jumbos* e mapeia se existem (e onde estão) defeitos, que costumam ter áreas pequenas e são geralmente pontuais.

O objetivo do problema estudado neste trabalho é gerar sequências de cortes guilhotinados (denominados padrões de corte) em placas de vidro planas de forma a minimizar a área total desperdiçada (isto é, pedaços de vidro cortados que não geram itens), respeitando um conjunto de restrições que é descrito a seguir.

2.1. Formalização do problema

Dado um conjunto de *jumbos* $\beta = \{b_1, \ldots, b_{|\beta|}\}$ de largura W e altura H, onde cada *jumbo* $b_j \in \beta$ possui um conjunto de defeitos D_{b_j}. Cada defeito $d \in D_{b_j}$ possui coordenadas de origem (x_d, y_d), largura w_d e altura h_d. Para cada par b_j, $b_l \in \beta$, a expressão $b_j <_{cut} b_l$ significa que o *jumbo* b_j deve ser utilizado no processo de corte antes de b_l.

Dado um conjunto de itens (também denominado de *batch*) $\mathcal{I} = \{1, \ldots, |\mathcal{I}|\}$, onde cada item $i \in \mathcal{I}$ possui largura w_i e altura h_i. Considere $S = \{S_1, ..., S_{|S|}\}$ como um conjunto de pilhas

[1]Página: http://www.roadef.org/challenge/2018/en/index.php

ordenadas, onde cada item $i \in \mathcal{I}$ pertence a uma única pilha $S_o \in S$. Para cada par de itens i e h em uma mesma pilha, a expressão $i <_{cut} h$ significa que o item i deve ser produzido no processo de corte antes do item h. Considere que não há relação de precedência de produção entre itens de pilhas diferentes.

Um padrão de corte consiste num conjunto de placas retangulares que compõem um *jumbo*. Uma placa no possui coordenadas (x_{no}, y_{no}) referentes a sua posição no *jumbo*, largura w_{no} e altura h_{no} e profundidade $depth_{no} = k$ que indica que a placa foi produzida com um *k-cut*.

Seja $P = \{p_1, p_2, \ldots, p_m\}$ uma solução viável, ou seja, um conjunto de $m \leq |\beta|$ padrões de corte que satisfazem todas as restrições do problema, no qual cada padrão de corte representa a sequência de cortes e produção de itens em um determinado *jumbo*. A área total de vidro desperdiçada é calculada através da soma da área desperdiçada de cada padrão de corte em P, sendo que a placa residual não utilizada à direita do último *1-cut* do padrão p_m não é considerada. Seja r_m o comprimento da área residual do padrão p_m, a função objetivo do problema pode ser calculada como

$$\min \; HWm - Hr_m - \sum_{i \in I} w_i h_i. \tag{1}$$

Adicionalmente, o problema possui o seguinte conjunto de restrições:

- Todos os cortes realizados nas placas devem ser guilhotinados;
- Os cortes de nível *1-cut* são sempre verticais, e é possível realizar cortes até o nível *3-cut* para geração de itens finais;
- Cortes *4-cut* são permitidos apenas para separar itens de desperdícios, não sendo possível utilizá-los para separar dois itens;
- Itens podem ser produzidos com menos de 3 cortes;
- Nenhum item produzido deve conter defeitos;
- Nenhum corte pode passar por cima de um defeito;
- Não é permitido violar a ordenação dos *jumbos*. As placas devem ser cortadas na ordem que são apresentadas;
- Não é permitido violar a ordenação de itens em uma pilha do *batch*;
- Os items podem ser rotacionados apenas em 90º, ou seja, eles podem estar em posição vertical ou horizontal nos padrões de corte;
- Todos os items do *batch* devem ser produzidos;
- Cada item do *batch* deve ser produzido apenas uma vez, ou seja, a superprodução de itens não é permitida;
- Os *jumbos* são sempre horizontais, e sua rotação não é permitida;
- A sobreposição de itens não é permitida;
- Dois cortes *1-cut* consecutivos devem respeitar uma largura mínima $wcut_{min}$ e não ultrapassar uma largura máxima $wcut_{max} < W$, com exceção de um corte que gera uma peça residual;
- Cada padrão de corte deve ter ao mínimo um *1-cut*;

https://proceedings.science/p/106878?lang=pt-br

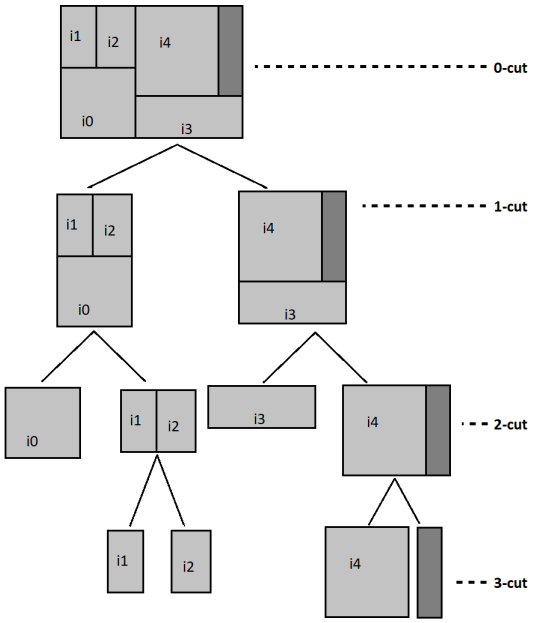
- Dois cortes *2-cut* consecutivos devem respeitar uma altura mínima $hcut_{min}$;
- A medida mínima de altura ou largura de um desperdício deve ser de $desp_{min}$.

Os valores $wcut_{min}$, $wcut_{max}$, $hcut_{min}$ e $desp_{min}$ são definidos pela entrada do problema.

2.2. Representação da solução

A fim de avaliar de forma eficiente as soluções geradas para o problema, é preciso adotar uma boa representação dos padrões de corte. Cada padrão de corte é representado por uma árvore (Dusberger e Raidl [2014]), e a solução formada por diversos padrões de corte como uma floresta. Para cada árvore, a raiz corresponde a placa inicial (o *jumbo*), e as folhas podem ser itens finais, desperdícios ou a área residual. Os filhos de um nó nível *k-cut* são obtidos através de cortes $(k+1)$-*cut*. A ordem de produção dos itens pode ser obtida através de uma busca em profundidade partindo do nó raiz de cada árvore. Assume-se que a raiz da árvore possui nível 0-*cut*. A Figura 2 exemplifica a representação em árvore de um padrão de corte.

Figura 2: Representação em árvore de um padrão de corte



3. Técnicas propostas

A fim de gerar soluções de boa qualidade para o problema, desenvolveu-se uma técnica que combina uma heurística gulosa randomizada para geração de soluções iniciais com um método de melhoria baseado em programação lógica por restrições. A heurística é apresentada na Subseção 3.1 e a modelagem de programação lógica por restrições utilizada no método de aprimoramento de soluções é apresentada na Subseção 3.2. A junção da heurística com o modelo para melhoria das soluções é apresentada em 3.3.

3.1. Heurística gulosa randomizada

A produção de itens em um *jumbo* é descrito em Algoritmo 1. O algoritmo recebe como parâmetros o *jumbo* b_j no qual os itens serão produzidos e o conjunto de itens de demandas dos clientes $\mathcal{I}$, onde existe uma pilha de placas a serem cortadas (linha 4). Uma placa é então selecionada para receber um item (linha 6). A seleção do item a ser disposto na placa é realizada

através de um critério guloso (linha 7). Se existir um item que caiba na área selecionada, levando em conta a presença de defeitos, o item é inserido na solução em uma posição (x, y) que não viole as restrições referentes a defeitos na placa (linhas 9-10). O corte guilhotinado é realizado gerando as duas placas geradas acima e à direita do item. Essas placas são inseridas na pilha de placas a serem cortadas (linhas 11-12) e o item produzido é removido do *batch* (linha 13). O algoritmo executa enquanto houver itens não produzidos e placas a serem cortadas. Ao final, é retornada a solução parcial produzida no *jumbo* contendo os itens produzidos no mesmo.

Dois critérios gulosos para seleção de itens são disponibilizados para a heurística. O primeiro critério consiste na seleção de um item i que maximize a equação $max(w_i, h_i)$, isto é, o item de maior dimensão no conjunto de itens disponíveis. O segundo critério guloso seleciona um item i em que dada uma área com largura w e altura h, minimiza a equação $w - w_i + h - h_i$, ou seja, o item com tamanho mais próximo da área a ser cortada. A heurística utiliza apenas um dos dois critérios gulosos para geração de uma solução. Na Subseção 3.3 é abordado como a alternância entre os critérios é utilizada.

A geração de soluções completas contendo toda a demanda do *batch* na quantidade de padrões de corte necessários é descrita em Algoritmo 2. O algoritmo recebe como parâmetros o conjunto de *jumbos* β e o conjunto de itens $\mathcal{I}$. Partindo de uma solução vazia (linha 2), ele entra em um laço de repetição, no qual enquanto existirem itens não produzidos (linha 4), o Algoritmo 1 é chamado para gerar um padrão de corte utilizando um *jumbo* em β (linhas 5-6). Ao final do laço de repetição, é retornado o conjunto de padrões de corte P que representa uma solução completa.

Algoritmo 1: CUT-JUMBO(b_i, $\mathcal{I}$)

```
1  itensProduzidos ← ∅;
2  placa ← {0, 0, W, H, jumbo};
3  pilha ← {placa};
4  while pilha ≠ ∅ and I ≠ ∅ do
5      node ← POP(pilha);
6      item ← GET-GREEDY-ITEM(I, node);
7      if item ≠ NULL then
8          {x_item, y_item} ← GET-ITEM-POSITION(item, node);
9          PUSH(itensProduzidos, {item, x_item, y_item});
10         PUSH(pilha, {x_item, y_item + h_item, w_item, h_node − h_item, jumbo});
11         PUSH(pilha, {x_item + w_item, y_item, w_node − w_item, h_node, jumbo});
12         I ← I \ {item};
13 return itensProduzidos;
```

3.2. Modelo de programação lógica por restrições

A complexidade e ineficiência de uma formulação de programação inteira para as restrições de corte guilhotinado, como apresentado em Pisinger e Sigurd [2007], motivou uma modelagem capaz de aprimorar soluções para o problema utilizando programação lógica por restrições. No presente trabalho desenvolveu-se um modelo com o objetivo de reorganizar os itens já dispostos em um *jumbo* para melhorar a utilização do espaço ocupado da placa, de forma que a área à direita do último *1-cut* seja a maior possível. Com isso, pretende-se obter mais espaço no *jumbo* que sirva para a produção de outros itens, ou no caso da última placa utilizada, aumentar a área residual a fim de diminuir o valor de desperdício da solução.

https://proceedings.science/p/106878?lang=pt-br

Algoritmo 2: GREEDY-HEURISTIC($\beta, \mathcal{I}$)

```
P ← ∅;
j ← 1;
while I ≠ ∅ do
    P ← P ∪ CUT-JUMBO(b_j,I);
    j ← j + 1;
return P;
```

Considere $\mathcal{I}_{b_j} \subseteq \mathcal{I}$ como o conjunto de itens em um dado *jumbo* $b_j \in \beta$. As cordenadas do $i \in \mathcal{I}_{b_j}$ ou defeito $d \in D_{b_j}$ no eixo x e y são denotadas por $x_i \in [0, W - min(w_i, h_i)]$ e $y_i \in [0, H - min(w_i, h_i)]$, respectivamente. Considere a variável $lim \in [0, W]$ que expressa a cordenada no eixo x do retângulo considerado como resíduo. Definem-se as variáveis de decisão:

$$rot_i = \begin{cases} 1, & \text{se o item i está rotacionado em 90º;} \\ 0, & \text{caso contrário.} \end{cases}$$

$$r_{ih} = \begin{cases} 0, & \text{se o item i está a esquerda do item h;} \\ 1, & \text{se o item i está abaixo do item h;} \\ 2, & \text{se o item i está a direita do item h;} \\ 3, & \text{se o item i está acima do item h.} \end{cases}$$

$$check = \begin{cases} 1, & \text{se a solução atual respeita as restrições de corte;} \\ 0, & \text{caso contrário.} \end{cases}$$

O problema pode ser modelado como:

$$\min lim \tag{2}$$

$$r_{ih} = 0 \wedge rot_i = 0 \rightarrow x_i + w_i \leq x_h \\ \wedge ((desp_{min} \geq x_h - (x_i + w_i)) \vee (0 = x_h - (x_i + w_i))) \; \forall i, h \in \mathcal{I}_{b_j}, \tag{3}$$

$$r_{ih} = 0 \wedge rot_i = 1 \rightarrow x_i + h_i \leq x_h \\ \wedge ((desp_{min} \geq x_h - (x_i + h_i)) \vee (0 = x_h - (x_i + h_i))) \; \forall i, h \in \mathcal{I}_{b_j}, \tag{4}$$

$$r_{ih} = 1 \wedge rot_i = 0 \rightarrow y_i + h_i \leq y_h \\ \wedge ((desp_{min} \geq y_h - (y_i + h_i)) \vee (0 = y_h - (y_i + h_i))) \; \forall i, h \in \mathcal{I}_{b_j}, \tag{5}$$

$$r_{ih} = 1 \wedge rot_i = 1 \rightarrow y_i + w_i \leq y_h \\ \wedge ((desp_{min} \geq y_h - (y_i + w_i)) \vee (0 = y_h - (y_i + w_i))) \; \forall i, h \in \mathcal{I}_{b_j}, \tag{6}$$

$$r_{ih} = 2 \wedge rot_h = 0 \rightarrow x_h + w_h \leq x_i \\ \wedge ((desp_{min} \geq x_i - (x_h + w_h)) \vee (0 = x_i - (x_h + w_h))) \; \forall i, h \in \mathcal{I}_{b_j}, \tag{7}$$

$$r_{ih} = 2 \wedge rot_h = 1 \rightarrow x_h + h_h \leq x_i \\ \wedge ((desp_{min} \geq x_i - (x_h + h_h)) \vee (0 = x_i - (x_h + h_h))) \; \forall i, h \in \mathcal{I}_{b_j}, \tag{8}$$

$$r_{ih} = 3 \wedge rot_h = 0 \rightarrow y_h + h_h \leq y_i \\ \wedge ((desp_{min} \geq y_i - (y_h + h_h)) \vee (0 = y_i - (y_h + h_h))) \; \forall i, h \in \mathcal{I}_{b_j}, \tag{9}$$

$$r_{ih} = 3 \wedge rot_h = 1 \rightarrow y_h + w_h \leq y_i \\ \wedge ((desp_{min} \geq y_i - (y_h + w_h)) \vee (0 = y_i - (y_h + w_h))) \; \forall i, h \in \mathcal{I}_{b_j}, \tag{10}$$

$$rot_i = 0 \rightarrow lim \geq x_i + w_i \; \forall i \in \mathcal{I}_{b_j}, \tag{11}$$

$$rot_i = 1 \rightarrow lim \geq x_i + h_i \; \forall i \in \mathcal{I}_{b_j}, \tag{12}$$

https://proceedings.science/p/106878?lang=pt-br

$$r_{ih} = 0 \wedge rot_i = 0 \rightarrow x_i + w_i \leq x_h \; \forall i \in \mathcal{I}_{b_j}, \; h \in D', \tag{13}$$
$$r_{ih} = 0 \wedge rot_i = 1 \rightarrow x_i + h_i \leq x_h \; \forall i \in \mathcal{I}_{b_j}, \; h \in D', \tag{14}$$
$$r_{ih} = 1 \wedge rot_i = 0 \rightarrow y_i + h_i \leq y_h \; \forall i \in \mathcal{I}_{b_j}, \; h \in D', \tag{15}$$
$$r_{ih} = 1 \wedge rot_i = 1 \rightarrow y_i + w_i \leq y_h \; \forall i \in \mathcal{I}_{b_j}, \; h \in D', \tag{16}$$
$$r_{ih} = 2 \wedge rot_h = 0 \rightarrow x_h + w_h \leq x_i \; \forall i \in \mathcal{I}_{b_j}, \; h \in D', \tag{17}$$
$$r_{ih} = 2 \wedge rot_h = 1 \rightarrow x_h + h_h \leq x_i \; \forall i \in \mathcal{I}_{b_j}, \; h \in D', \tag{18}$$
$$r_{ih} = 3 \wedge rot_h = 0 \rightarrow y_h + h_h \leq y_i \; \forall i \in \mathcal{I}_{b_j}, \; h \in D', \tag{19}$$
$$r_{ih} = 3 \wedge rot_h = 1 \rightarrow y_h + w_h \leq y_i \; \forall i \in \mathcal{I}_{b_j}, \; h \in D', \tag{20}$$
$$check = 1. \tag{21}$$

A função objetivo (2) minimiza a posição do último 1-cut do jumbo atual b_j, o que delimita a área de inicio do resíduo no último *jumbo* da solução. As restrições (3)-(10) garantem que não exista sobreposição e que o corte mínimo entre dois itens seja respeitado. As restrições (11)-(12) obrigam a variável lim a estar sempre após o item alocado mais a direita de b_j. As restrições (13)-(20) garantem que um item não seja alocado sobre um defeito. A restrição (21) assegura que a variável $check$ é igual a um, isto é, o modelo é inicializado assumindo que existe uma solução viável. Uma função de validação de solução foi implementada para cada possível solução gerada pelo modelo. A solução é validada em relação a presença de cortes guilhotinados e a relação de precedência dos itens produzidos. A validação altera o valor da variável $check$ para 0 caso a possível solução encontrada não seja viável em relação às restrições do problema, e para 1 caso contrário.

3.3. Método de geração e melhoria de soluções

A geração de soluções para o problema consiste de um método *multi-start* utilizando a heurística gulosa randomizada. Um método de melhoria das soluções geradas é aplicado combinando o modelo lógico por restrições e a heurística. O método *multi-start* seleciona a solução viável encontrada pela heurística com melhor valor dentre todas as geradas, sendo que a cada 1000 soluções produzidas o critério guloso é alternado entre as duas opções existentes. Em seguida, para cada *jumbo* da melhor solução encontrada, é executado o modelo de programação lógica por restrições a fim de reorganizar a posição dos itens da placa. Caso seja obtido um espaço maior à direita do último *1-cut*, a heurística gulosa é chamada para este espaço, a fim de tentar dispor itens que tenham sido produzidos em placas posteriores ao *jumbo* corrente. Este processo é repetido até a última placa, onde o modelo é chamado para aumentar a área residual a fim de diminuir o valor da solução final.

4. Experimentos computacionais

Foram realizadas duas baterias de teste com dois grupos de instâncias disponíveis na página do *ROADEF/EURO Challenge 2018: Cutting Optimization Problem*. A primeira bateria executou com tempo limite de 180 segundos para cada instância, divididos em 70 segundos para o *multi-start* e 110 segundos para o método de melhoria. A segunda bateria teve tempo limite de 3600 segundos para cada instância, sendo 1800 segundos para execução do *multi-start* e 1800 segundos para o método de melhoria. O tempo limite do modelo de programação lógica por restrições foi dividido igualmente para cada *jumbo* da solução passada como entrada. Em ambos os conjuntos de teste foram utilizados os grupos A, com 20 instâncias, e B, com 15 instâncias. O grupo A possui instâncias pequenas e médias, com quantidade de itens por instância variando entre 5 e 392, e o grupo B possui instâncias médias e grandes, com quantidade de itens por instância variando entre 68 e 656.

Os experimentos computacionais foram realizados utilizando-se uma máquina Dell OptiPlex7040, com processador Intel(R) Core(TM) i5-6500 CPU @ 3.20GHz × 4, 8GB de memória,

https://proceedings.science/p/106878?lang=pt-br

utilizando o sistema operacional Ubuntu. Para o modelo de programação lógica por restrições foi utilizada a ferramenta IBM ILOG CP Optimizer Optimization Studio 12.8. Os resultados dos experimentos com tempo limite de 180 segundos e 3600 segundos estão apresentados nas Tabelas 1 e 2, respectivamente. A coluna Solução de ambas as tabelas apresenta o valor da solução utilizando a equação (1). As duas colunas subsequentes apresentam a porcentagem de área desperdiçada em relação a área total das placas utilizadas, e a porcentagem de área desperdiçada dos melhores resultados do *ROADEF/EURO Challenge 2018: Cutting Optimization Problem*, disponível na página do desafio.

Foi observado também que com os testes de 3600 segundos, houveram melhorias no valor da solução de 17 das 20 instâncias do grupo A, e 10 das 15 instâncias do grupo B. Para uma das 20 instâncias do grupo A o melhor resultado encontrado no *ROADEF/EURO Challenge 2018: Cutting Optimization Problem* foi alcançado. Também é importante pontuar que para o grupo A de instâncias, todas as soluções tiveram o mesmo número de *jumbos* utilizados que os melhores resultados encontrados, garantindo resultados satisfatórios. Os resultados apresentados garantiram a classificação para a fase final do *ROADEF/EURO Challenge 2018: Cutting Optimization Problem*.

Tabela 1: Resultados dos testes de 180 segundos

Instância	Solução	% Desp.	% Melhor	Instância	Solução	% Desp.	% Melhor
A1	425,486	2.2%	2.2%	A19	8,925,194	15.4%	6.4%
A2	14,690,819	12.7%	6.8%	A20	1,622,005	8.4%	7.6%
A3	7,922,700	13.7%	6.0%	B1	9,614,178	9.9%	3.3%
A4	9,630,420	16.6%	6.8%	B2	74,156,945	18.3%	4.8%
A5	11,732,373	15.2%	5.6%	B3	86,692,493	19.5%	5.6%
A6	8,272,050	14.3%	5.7%	B4	41,418,715	21.5%	4.8%
A7	17,392,890	18.0%	6.0%	B5	168,455,615	33.6%	17.8%
A8	30,133,124	17.3%	6.9%	B6	60,112,447	22.2%	5.7%
A9	9,639,606	16.6%	6.3%	B7	33,284,679	14.4%	4.5%
A10	14,388,481	14.9%	6.2%	B8	94,074,169	21.2%	4.9%
A11	18,845,659	19.5%	6.5%	B9	77,736,277	20.1%	5.6%
A12	5,032,174	13.0%	5.8%	B10	108,021,263	23.3%	6.7%
A13	58,322,313	21.6%	6.0%	B11	91,217,390	20.5%	6.9%
A14	54,739,158	18.9%	7.9%	B12	75,160,567	21.6%	5.9%
A15	67,068,101	21.7%	7.7%	B13	142,801,605	22.4%	5.5%
A16	7,935,323	13.7%	7.4%	B14	44,720,680	19.3%	4.9%
A17	6,336,121	16.4%	9.4%	B15	112,782,401	20.1%	6.0%
A18	15,229,938	19.7%	7.4%				

5. Comentários finais

Neste trabalho foi estudado um problema restrito de corte bidimensional guilhotinado, com possibilidade de rotação de itens e presença de placas defeituosas. O estudo apresentado neste trabalho propõe uma técnica mista que combina uma heurística gulosa randomizada e uma modelagem de programação lógica por restrições a fim de gerar e aprimorar soluções.

A modelagem de programação lógica por restrições apresentou uma boa performance dada uma solução inicial recebida como entrada. Na maioria dos testes, cada *jumbo* da solução dada como entrada para a técnica retornava a melhor configuração de posição dos itens previamente posicionados em até 20 segundos. A quantidade de *jumbos* das soluções finais geradas também foi muito próxima da quantidade usada nas melhores soluções encontradas no *challenge*. Para as instâncias do grupo A, o número de placas utilizadas foi igual ao das melhores soluções, enquanto para o grupo B houve uma diferença de até 3 placas.

Visto que o resultado final da técnica de melhoria de soluções depende diretamente da seleção de itens a serem dispostos em cada *jumbo*, estuda-se como trabalhos futuros a utilização de novas técnicas que realizem diferentes escolhas de itens para corte em cada placa. Dentre essas

Tabela 2: Resultados dos testes de 3600 segundos

Instância	Solução	% Desp.	% Desp. Melhor	Instância	Solução	% Desp.	% Desp. Melhor
A1	425,486	2.2%	2.2%	A19	7,031,294	12.2%	6.4%
A2	12,023,309	10.4%	6.8%	A20	1,622,005	8.4%	7.6%
A3	7,059,210	12.2%	6.0%	B1	8,647,968	8.9%	3.3%
A4	9,148,920	15.8%	6.8%	B2	71,781,545	17.7%	4.8%
A5	10,310,343	13.4%	5.6%	B3	82,474,553	18.6%	5.6%
A6	7,540,170	13.0%	5.7%	B4	38,776,885	20.1%	4.8%
A7	13,730,280	14.2%	6.0%	B5	168,455,615	33.6%	17.8%
A8	28,078,724	16.2%	6.9%	B6	53,750,227	21.4%	5.7%
A9	7,902,996	13.7%	6.3%	B7	33,284,679	14.4%	4.5%
A10	13,762,531	14.3%	6.2%	B8	87,615,649	19.8%	4.9%
A11	17,629,069	18.3%	6.5%	B9	77,736,277	20.1%	5.6%
A12	4,374,124	11.4%	5.8%	B10	100,211,333	21.7%	6.7%
A13	49,883,223	18.5%	6.0%	B11	91,217,390	20.5%	6.9%
A14	54,215,928	18.7%	7.9%	B12	71,790,067	20.7%	5.9%
A15	63,793,901	20.7%	7.7%	B13	142,801,605	22.4%	5.5%
A16	7,155,293	12.4%	7.4%	B14	42,149,470	18.2%	4.9%
A17	6,336,121	16.4%	9.4%	B15	110,156,621	19.7%	6.0%
A18	13,926,678	18.1%	7.4%				

novas técnicas, é possível citar novas heurísticas gulosas que realizem diferentes combinações de corte que a heurística atual não consiga executar, como a possibilidade de produzir itens lado a lado em um único *1-cut*, e um modelo de geração de colunas que permita selecionar diferentes itens.

Agradecimentos: O trabalho de Rafael Melo foi financiado pela Universidade Federal da Bahia, pelo Ministério de Ciência, Tecnologia, Inovação e Comunicação pela Fundação de Amparo à Pesquisa do Estado da Bahia (FAPESB) e pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPQ). O trabalho de Michell Queiroz foi financiado por uma bolsa CAPES. O trabalho de Junot foi financiado por uma bolsa FAPESB. O trabalho de Luis Modesto foi financiado por uma bolsa UFBA/MCTIC. E o trabalho de Mateus Carvalho foi financiado por uma bolsa PIBIC/UFBA-CNPQ.

References

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.

Beasley, J. (2004). A population heuristic for constrained two-dimensional non-guillotine cutting. *European Journal of Operational Research*, 156(3):601–627.

Dolatabadi, M., Lodi, A., e Monaci, M. (2012). Exact algorithms for the two-dimensional guillotine knapsack. *Computers & Operations Research*, 39(1):48–53.

Dusberger, F. e Raidl, G. R. (2014). A variable neighborhood search using very large neighborhood structures for the 3-staged 2-dimensional cutting stock problem. In *International Workshop on Hybrid Metaheuristics*, p. 85–99. Springer.

Furini, F., Malaguti, E., Durán, R. M., Persiani, A., e Toth, P. (2012). A column generation heuristic for the two-dimensional two-staged guillotine cutting stock problem with multiple stock size. *European Journal of Operational Research*, 218(1):251–260.

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

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.

Silva, E., Alvelos, F., e de Carvalho, J. V. (2010). An integer programming model for two-and three-stage two-dimensional cutting stock problems. *European Journal of Operational Research*, 205 (3):699–708.

Velasco, A. S. e Barboza, E. U. (2014). Geraçao de padroes de corte bidimensionais guilhotinados via grasp. *Proceedings of XLVI SBPO, Salvador*, p. 1–12.

Wuttke, D. A. e Heese, H. S. (2018). Two-dimensional cutting stock problem with sequence dependent setup times. *European Journal of Operational Research*, 265(1):303–315.