São José dos Campos - SP
LV SBPO
LV Simpósio Brasileiro de Pesquisa Operacional
São José dos Campos, SP - 6 a 9 de novembro de 2023

METAHEURÍSTICA PARA RESOLUÇÃO DO PROBLEMA INTEGRADO DE CORTE DE PEÇAS IRREGULARES E DETERMINAÇÃO DO CAMINHO DE CORTE

Leonardo Trevisan, Marina Andretta
Universidade de São Paulo
Rua Trabalhador São-carlense, 400, São Carlos - SP
leotrevisan@usp.br, andretta@icmc.usp.br

Larissa Tebaldi Oliveira
Departamento de Matematica, Universidade Estadual Paulista Júlio de Mesquita Filho
Av. Eng. Luiz Edmundo Carrijo Coube, 14-01, Bauru - SP
larissa.tebaldi@unesp.br

RESUMO

Problemas de corte de peças irregulares em faixa são problemas que tem como objetivo determinar um leiaute para as peças dentro de um recipiente de altura fixada e comprimento ilimitado, de modo a utilizar o menor comprimento possível. Este problema tem grande importância prática uma vez que surge em vários processos industriais. Uma vez construído o leiaute, em muitos casos é preciso definir como cortar de forma ótima as peças do recipiente, otimizando o tempo e/ou os movimentos de corte. Este problema é conhecido como problema de determinação do caminho de corte, e também tem grande relevância prática. Neste trabalho, propomos um método heurístico para resolver ambos os problemas de forma integrada. O método proposto se mostrou promissor, conseguindo bons resultados tanto para otimização individual dos problemas, quanto em conjunto.

PALAVRAS CHAVE. corte de peças irregulares, determinação do caminho de corte, metaheurística.

Tópicos: MH – Metaheurísticas, POI – PO na Indústria

ABSTRACT

Irregular strip packing problems aim to determine a layout for pieces within a container with a fixed height and unlimited length, in order to minimize the utilized lenght. This problem is of great practical importance as it arises in inumerous industrial processes. After the layout is built, it is frequently required to identify the most effective cutting path for the pieces within the conteiner, optimizing cutting time and/or cutting moves. This problem is known as the cutting path determination problem and also has significant practical relevance. In this work, we propose a heuristic method to solve both problems in an integrated way. The proposed method has shown promising results for both individual problem optimization and integrated optimization

KEYWORDS. irregular strip packing problem, cutting path determination problem, metaheuristics.

Paper topics: MH – Metaheuristics, POI – OR in Industry

1. Introdução

Muitos produtos que utilizamos diariamente são produzidos a partir de várias partes ou peças que foram cortadas de objetos maiores (recipientes). Aqui, surge a necessidade de buscar um leiaute eficiente para o corte dessas peças, que minimize o desperdício de material. Esse problema de otimização combinatória é conhecido na literatura como o problema de corte e empacotamento.

Na prática, em algumas indústrias, profissionais experientes resolvem esse problema posicionando as peças nos recipientes, de maneira manual, buscando reduzir ao máximo as perdas de material. Em alguns casos, esses profissionais contam com o auxílio de softwares comerciais especializados em criar leiautes. Devido à sua importância tanto do ponto de vista ambiental quanto econômico, juntamente à sua complexidade de resolução, os problemas de corte e empacotamento têm atraído o interesse de pesquisadores da área de pesquisa operacional ao longo das últimas décadas (Bennell e Oliveira [2009], Álvarez-Valdés et al. [2018], Leão et al. [2020]).

Após a definição de um leiaute para as peças, surge a necessidade de determinar a melhor maneira de cortá-las. Na literatura, esse problema é conhecido como o problema de determinação do caminho de corte (Manber e Israni [1984], Moreira et al. [2007], Usberti et al. [2011], Silva et al. [2015] e Silva et al. [2019]). Esse problema busca a definição da trajetória que uma ou mais ferramentas de corte devem seguir para executar um leiaute dado. O objetivo pode ser, por exemplo, reduzir o tempo total de corte, permitindo assim a produção de um maior número de leiautes ao final de um turno de trabalho. Outro objetivo bastante comum deste tipo de problema é buscar reduzir o comprimento do caminho de corte, especialmente quando a ferramenta tem um custo elevado e se desgasta ao longo do processo de corte. Um caminho de corte mais curto pode resultar em um menor consumo de energia elétrica, contribuindo diretamente para a preservação dos recursos ambientais. Portanto, otimizar o caminho de corte não apenas traz benefícios econômicos, mas também impacta positivamente a sustentabilidade.

Desde a década de 1980, diversos autores têm ressaltado que as decisões tomadas durante a etapa de empacotamento exercem uma forte influência na resolução do problema do caminho mínimo de corte (Manber e Israni [1984], Sherif et al. [2014], Anand e Babu [2015], Oliveira et al. [2020]). Ou seja, nem sempre os leiautes de menor comprimento resultam nos caminhos de corte mais curtos. Em muitos casos, leiautes menos eficientes em termos de empacotamento, ou seja, menos compactos, podem levar a caminhos de corte mais reduzidos. Pesquisas anteriores demonstraram que é vantajoso abordar os problemas de empacotamento de peças irregulares em faixas e determinação do caminho mínimo de corte de forma integrada (Oliveira [2019], Oliveira et al. [2020]).

Neste artigo, propomos um método baseado no algoritmo genético de chaves aleatórias viciadas (*Biased Random-Key Genetic Algorithm*) para a resolução conjunta dos problemas de empacotamento e caminho mínimo de corte. Este texto está estruturado da seguinte maneira. Na Seção 2 definimos o problema de empacotamento e caminho mínimo de corte, na Seção 3 descrevemos o método proposto para a resolução deste problema e, na Seção 4, reportamos os testes computacionais realizados. Por fim, algumas considerações finais sobre o trabalho e as perspectivas de trabalho futuro são apresentadas na Seção 5.

2. Definição dos problemas

O problema do corte e empacotamento de peças irregulares em faixa busca empacotar peças de formatos irregulares, sejam elas convexas ou não-convexas, em um recipiente retangular com altura fixa e comprimento ilimitado. O objetivo é minimizar o comprimento total utilizado do recipiente, buscando otimizar o aproveitamento do espaço disponível. Na Figura 1 é exemplificado um leiaute obtido para uma empresa especializada na fabricação de roupas de banho.

Figura 1: Um plano de corte para roupas de banho

Fonte: Bennell e Oliveira [2008]

O ponto crucial deste problema é respeitar as seguintes restrições: (i) não permitir a sobreposição de peças; (ii) alocar integralmente as peças dentro da placa; e (iii) garantir o corte de todas as peças exigidas.

De acordo com Wäscher et al. [2007], o problema de empacotamento de peças irregulares, também conhecido como *nesting*, é um problema bidimensional e irregular. Sua particularidade, assim como seu principal desafio, reside na necessidade do tratamento geométrico das peças. As principais dificuldades computacionais associadas à geometria das peças são determinar automaticamente se duas peças se sobrepõem ou não, e garantir que elas estejam completamente contidas na placa. Bennell e Oliveira [2008] discutem as estruturas mais comumente utilizadas para representar a parte geométrica deste problema.

Segundo Fowler et al. [1981], o problema de corte e empacotamento de peças irregulares é classificado como NP-completo. Isso justifica o cenário atual, em que há uma abundância de artigos que empregam heurísticas e/ou metaheurísticas para abordar o problema. Em Bennell e Oliveira [2009], Álvarez-Valdés et al. [2018] e Leão et al. [2020], é possível encontrar tutoriais sobre as diferentes abordagens de resolução propostas na literatura para esse problema.

Depois de definido o empacotamento das peças, surge o problema adicional de encontrar o melhor modo de cortar as peças. Na literatura, esse problema é conhecido como o problema de determinação do caminho de corte (*Cutting Path Determination Problem - CPDP*) e consiste em definir uma trajetória que uma ou mais ferramentas de corte devem seguir para executar um determinado leiaute.

De modo geral, existem duas formas de cortar as peças: o corte por peças e o corte por arestas. Na primeira abordagem, o corte é realizado a partir de um dos vértices da peça. Em outras palavras, a ferramenta de corte inicia e termina seu corte em um vértice específico, realizando o corte completo da peça em uma única vez. Após concluir o corte de uma peça, a ferramenta avança para a próxima peça ou retorna à origem se todas as peças já foram cortadas. Durante esse processo, a ferramenta de corte pode ser temporariamente desligada para se movimentar sobre as peças ainda não cortadas.

Na segunda abordagem do problema, o corte é realizado por arestas, ou seja, cada peça é considerada como um conjunto de arestas a serem cortadas, não necessariamente em sequência. Nesta versão do problema também é permitido que a ferramenta de corte possa ser temporariamente desligada para se movimentar sobre as peças ainda não cortadas (movimentos ociosos).

Neste trabalho, consideramos a versão do corte por arestas para o problema de determinar o caminho de corte. Consideramos também a divisão das arestas obrigatórias em arestas menores

sempre que um vértice de outra aresta estiver sobre ela, conforme proposto por Silva et al. [2019]. Estamos supondo que o corte feito não causa muito desgaste (então as peças podem ficar encostadas no leiaute) e que a ferramenta de corte possuí apenas uma cabeça de corte.

Assim, neste trabalho, estamos interessados em, dado um conjunto de itens irregulares, representados por polígonos convexos ou não-convexos, e um recipiente com altura fixa e comprimento ilimitado, definir o posicionamento dos itens no recipiente, sem sobreposição. Depois disso, usando o corte por arestas, desejamos definir um caminho de corte para o leiaute construído. Idealmente, gostaríamos de obter o leiaute com menor comprimento e o menor caminho de corte possíveis. No entanto, em alguns casos, esses objetivos podem ser conflitantes.

3. Método de solução

Para resolver o problema de determinação do leiaute integrado ao problema de determinar o caminho de corte, resolvemos cada um destes sub-problemas separadamente e depois usamos uma metaheurística para integrá-los. O problema de determinação do leiaute é resolvido por uma variação da heurística construtiva *Bottom-left*, como apresentamos na Seção 3.1. O problema de determinação do caminho de corte é modelado como um problema de otimização linear inteiro e revolvido pelo *solver* Gurobi (`http://gurobi.com`), como descrito na Seção 3.2. Por fim, a integração dos dois sub-problemas é feita pelo algoritmo genético de chaves aleatórias viciadas (*Biased Random-Key Genetic Algorithm* - BRKGA), como pode ser visto na Seção 3.3.

3.1. Resolução do problema de determinação do leiaute

Existem muitas heurísticas construtivas na literatura que podem ser usadas para resolver um problema de empacotamento de peças irregulares em faixa Bennell e Oliveira [2008]. Uma das mais conhecidas e usadas é a heurística *Bottom-left*. A ideia deste método consiste em, dada uma lista de peças em uma certa ordem, inserir as peças no recipiente uma a uma, sempre as colocando na posição mais à esquerda e abaixo possível de cada iteração. Ou seja, como nosso leiaute é horizontal e queremos minimizar o comprimento total gasto, procuramos sempre a posição mais a esquerda possível e, como critério de desempate, escolhemos a mais abaixo dentre as opções. É importante notar que diferentes ordenações de peças geram leiautes diferentes.

Já mencionamos que consideramos que todas as peças são representadas por conjuntos de polígonos convexos. Além disso, supomos que cada peça tem um vértice de referência. Esse vértice deve ser posicionado no recipiente de forma a garantir que cada peça esteja inteiramente contida no recipiente, que não haja sobreposição entre peças e que, a cada iteração da heurística *Bottom-letf*, a peça seja inserida na posição mais à esquerda e abaixo disponível. Para melhorar o desempenho computacional, também supomos que peças iguais são representadas uma única vez, mas que cada peça possui uma quantidade de cópias disponível.

Consideramos também que o recipiente está posicionado no plano cartesiano, com seu canto inferior esquerdo posicionado em $(0, 0)$. Para garantir a contenção das peças no recipiente, utilizamos o conceito de IFP (*Inner-Fit Polygon*) Bennell e Oliveira [2008]. Neste caso, o IFP de uma peça i é dado pelo retângulo que contém todos os pontos do recipiente em que o ponto de referência de i pode ser inserido de forma que a peça fique inteiramente contida no recipiente.

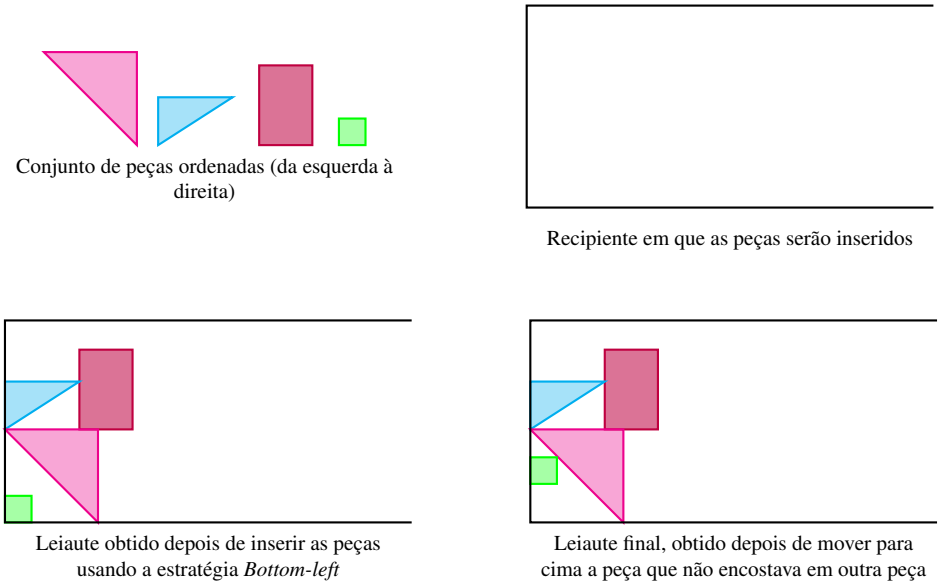
Para garantir que duas peças i e j não se sobreponham, utilizamos o conceito de NFP (*No-Fit Polygon* Bennell e Oliveira [2008]. O NFP_{ij} é um polígono, construído em uma fase de pré-processamento, a partir dos polígonos de i e j. Se o ponto de referência de j está no interior do NFP_{ij}, as peças i e j se sobrepõem; se está na borda do NFP_{ij}, as peças i e j se encostam; caso contrário, as peças estão separadas.

Para calcular o ponto mais à esquerda e abaixo para inserir uma dada peça i no recipiente, também utilizamos os conceitos de IFP e NFP. Note que, para uma peça ficar em uma posição mais

à esquerda e abaixo, ela necessariamente encosta em duas bordas do recipiente, ou em uma borda do recipiente e em uma peça já alocada, ou em duas peças já alocadas. Por isso, para descobrir o ponto em que devermos inserir a peça i, calculamos a interseção de todas as arestas do IFP da peça i e de todas as arestas dos NFP$_{ji}$, para toda peça j já alocada. Depois disso, excluímos todos os pontos que estão fora do IFP da peça i e todos os que estão no interior de algum NFP$_{ji}$ (com j peça já alocada). Com isso, temos a lista de todos os pontos em que podemos alocar o ponto de referência de i de forma que a peça i encoste em bordas ou peças já alocadas. Dentre todos esses pontos, escolhemos o que tem o menor valor na coordenada x (ou seja, o mais à esquerda). Em caso de empate, escolhemos o que tem o menor valor na coordenada y (ou seja, o mais abaixo).

Depois de calculado o leiaute com a heurística *Bottom-left*, fazemos um pós-processamento. Já sabemos que todas as peças são inseridas de modo a encostrar ou nas bordas do recipiente ou em outras peças. Caso, no leiaute definido, alguma peça esteja encostando apenas nas bordas do recipiente (e isso pode acontecer apenas na borda da esquerda e de baixo, pelo desenho da heurística), podemos, sem mudar o comprimento do leiaute, subir essas peças de forma que elas encostem em outras peças. Isso também é feito usando os NFPs. A Figura 2 ilustra o leiaute obtido pela heurística *Bottom-letf* para um conjunto ordenado de quatro peças, incluindo a etapa de pós-processamento para que todas as peças se toquem. Daqui em diante, quando nos referirmos à heurística *Bottom-left*, estamos nos referindo à aplicação da heurística seguida deste pós-processamento.

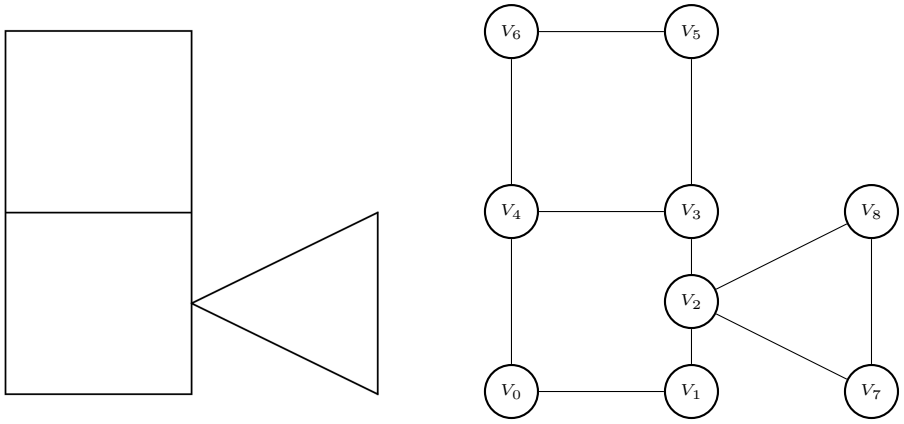
Figura 2: Exemplo de um leiaute construído com a heurística *Bottom-left*



3.2. Resolução do problema de determinação do caminho de corte

Neste trabalho, o modelo utilizado para descrever o problema da determinação do caminho de corte (CPDP) é o mesmo encontrado em Silva et al. [2019] e é baseado na modelagem de Christofides et al. [1981]. Para construir este modelo, primeiramente o leiaute é transformado em um grafo correspondente, em que os vértices e arestas das peças se transformam em vértices e arestas dos grafos, mas são eliminadas repetições e as arestas das peças podem ser divididas caso algum outro vértice esteja posicionado nelas. A Figura 3 mostra um exemplo de leiaute e o respectivo grafo.

Figura 3: Exemplo de transição de um leiaute para um grafo correspondente



(a) Exemplo de leiaute (b) Grafo correspondente ao leiaute

Note que, pela construção do leiaute, em que as peças sempre se encostam, esse grafo tem sempre apenas uma componente conexa. Chamamos de G este grafo, V o conjunto de seus vértices e de E_R o conjunto destas arestas do grafo, que correspondem às arestas que devem ser cortadas. Acrescentamos a este grafo o conjunto de todas as arestas que ligam quaisquer vértices do grafo (incluindo as arestas de E_R, que são duplicadas), obtendo um conjunto de arestas que chamamos de E. Estas novas arestas correspondem aos movimentos ociosos, e não precisam necessariamente ser usados.

Com este grafo, temos então um Problema do Carteiro Rural (*Rural Postman Problem* - RPP). Ele é definido em um grafo $G = (V, E)$, no qual V representa o conjunto de vértices e E o conjunto de arestas. Cada aresta $e = (i, j)$ possui um custo c_{ij} associado a ela (no nosso caso, a distância euclidiana entre os vértices). Além disso, é exigido que um conjunto E_R de arestas faça parte da solução, ou seja, que todas as arestas desse conjunto sejam percorridas pelo trajeto da solução. Deste modo, o RPP busca a determinação de um caminho de custo mínimo que atravessa todas as arestas de E_R ao menos uma vez. Sabemos que o problema é NP-díficil (Lenstra e Kan [1976]), no entanto, quando trabalhamos com grafos nos quais o conjunto E_R é conexo (que é o nosso caso), o problema acaba sendo reduzido e resolvido em tempo polinomial (Silva et al. [2019]).

Dados os parâmetros:

- $G = (V, E)$, grafo G composto pelo conjunto de vértices V e pelo conjunto de arestas E;
- E_R, conjunto de arestas obrigatórias pertencentes a E;
- c_e, custo associado a aresta e;
- $\delta(v_i)$, conjunto de arestas que possuem uma extremidade no vértice v_i;

e as variáveis:

- x_e com valor 1 se a aresta e é atravessada pela solução, 0 caso contrário;
- w_{v_i}, inteira, utilizada para garantir a cardinalidade par do vértice v_i;

o modelo é dado por

$$\min \sum_{e \in E} c_e x_e \tag{1}$$

$$s.a : |E_R \cap \delta(v_i)| + \sum_{e \in \delta(v_i)} x_e = 2w_{v_i}, \qquad \forall i \in V, \tag{2}$$

$$x_e \in \{0, 1\}, \qquad \forall e \in E, \tag{3}$$

$$w_{v_i} \in \mathbb{Z}_+, \qquad \forall v_i \in V. \tag{4}$$

A função objetivo (1) soma os custos das arestas atravessadas (com valor $x_e = 1$) e busca minimizá-lo. As restrições (2) garantem a cardinalidade par dos vértices pertencentes à solução. Isso é feito com o objetivo de garantir que a máquina tenha entrado e saído de cada vértice um número igual de vezes e, consequentemente, seja possível criar um ciclo que percorra todos os vértices e arestas e retorne ao mesmo local de origem. As últimas duas restrições (3) e (4) definem os domínios das variáveis.

Portanto, o que o modelo busca encontrar é um conjunto E_O de arestas correspondentes aos movimentos ociosos que faça com que todos os vértices do grafo $\bar{G} = (V, E_R \cup E_O)$ tenham grau par, o que garante que haja um circuito euleriano em $\bar{G}$.

Assim, para calcular o caminho de corte, dado um leiaute, transformamos o leiaute em um grafo G (como descrito anteriormente), resolvemos o modelo $(1) - (4)$ para encontrar o conjunto de arestas E_O e, usando essas arestas, determinamos o caminho de corte encontrando um circuito euleriano em $\bar{G}$.

3.3. Resolução do problema integrado

Para a resolução do problema integrado utilizamos o algoritmo genético de chaves aleatórias viciadas (*Biased Random-Key Genetic Algorithm*), conhecido como BRKGA. O BRKGA deste artigo é baseado no trabalho Mundim et al. [2017]. A ideia por trás deste algoritmo é gerar uma população de P indivíduos (cromossomos) que representam soluções do problema abordado. No BRKGA, a população é ranqueada de acordo com sua qualidade e dividida de modo que os indivíduos que possuam as melhores soluções formem a chamada população elite P_e, de tamanho previamente fixado.

Operações de mutação e cruzamento entre indivíduos da elite e da população geral são realizadas a fim de gerar novos indivíduos, ou seja, novas soluções, que são acrescidas à população. Estes cruzamentos consistem em operações enviesadas entre 2 cromossomos, um da população elite P_e, e outro da população geral, criando uma novo indivíduo a partir deles. Ela é enviesada pois a chance α do novo cromossomo herdar informações da sequência da elite é sempre igual superior a $0,50$. Após feitas as operações de mutação e cruzamento, e com os novos integrantes acrescidos à população, é feito um novo ranqueamento das soluções. Novamente, os indivíduos de melhor qualidade irão compor a população elite e aqueles que apresentarem resultados de pouca qualidade e ficarem ranqueados em posições inferiores ao tamanho fixado da população serão eliminados.

Para adaptar o BRKGA ao nosso problema precisamos: i) definir como serão nossos cromossomos e ii) como avaliaremos sua qualidade. Como pretendíamos utilizar a *Bottom-left* para a construção dos leiautes, e ela depende da ordem de inserção das peças, definimos o cromossomo como um vetor de n posições, em que n é o número de peças da instância. Cada posição do vetor cromossomo representa o ID do i-ésimo item inserido.

Assim, a operação de cruzamento fica definida por n escolhas aleatórias entre os 2 cromossomos pais, e cada escolha, acrescentamos o primeiro índice não utilizado da sequência escolhida

para uma nova sequência que, no final, terá n itens e constituirá um leiaute. Como no BRKGA a operação cruzamento é enviesada, a chance α do novo cromossomo herdar informações da sequência da elite é sempre igual superior a $0,50$.

Já para a operação de mutação, definimos que, para cada rodada, escolhemos um indivíduo de P_e e outro de P_{e^-} (indivíduos da população que não pertencem à elite) e realizamos uma simples troca de posição entre um número fixo de itens de cada ordem. Em nosso caso, fixamos esse número em 5 para todos os casos.

A fim de gerar uma população inicial, geramos sequências aleatórias de cromossomos e, junto delas, acrescentamos mais 4 ordens específicas para todos os casos: ordem crescente e decrescente da área dos itens, e ordem crescente e decrescente da área do menor retângulo que cobre cada item. Isso foi motivado pela ideia de que escolher ordens como essa podem ser consideradas heurísticas simples e triviais para se resolver problemas de empacotamento.

Para avaliar a qualidade de um cromossomo precisamos transformá-lo em uma solução do problema e para isso utilizamos uma função de decodificação que, neste trabalho, é composta pela a heurística construtiva *Bottom-left*, apresentada na Seção 3.1, e pelo modelo linear descrito na Seção 3.2. No BRKGA, cada indivíduo é associado o seu valor da Função *fit*, que é o valor qualidade utilizado para avaliar a solução associada a cada cromossomo. Aqui, ora ela é representada pelo valor do comprimento horizontal do leiaute associado ao cromossomo (que chamamos de fit_1), ora pelo valor do caminho de corte (que chamamos de fit_2) e, em um terceiro momento, utilizamos uma soma ponderada entre esses valores com o intuito de achar uma solução que tenta otimizar os dois problemas ao mesmo tempo (que chamamos de fit_3, mais detalhada na Seção 4).

4. Experimentos numéricos

Nesta seção, apresentamos os resultados computacionais que foram obtidos em nossos experimentos, bem como os parâmetros utilizados para o BRKGA e os detalhes da implementação de cada rotina. Todos os códigos utilizados para compor nosso método de solução foram escritos na linguagem Python (3.9.16) e para a rotina que resolve o CPDP, utilizamos o *solver Gurobi* (10.0.1). Utilizamos também as bibliotecas *Numpy* e *Matplotlib* para a realização de cálculos e visualização dos leiautes e grafos gerados, respectivamente. Os testes foram realizados em um Intel® Core™ i5-9300, 2.40GHz com 8GB de RAM no sistema operacional Windows 10.

Os parâmetros utilizados no BRKGA estão exibidos na Tabela 1. O parâmetro α foi escolhido por conta dos resultados observados em Mundim et al. [2017], enquanto os valores de P e do número de gerações foram decididos com base em testes de tempo para cada instância. Por se tratar de um método heurístico e não uma solução exata, ajustamos os parâmetros de modo que a execução da instância mais longa não ultrapassasse 15 minutos, simulando o que seria uma aplicação prática.

Parâmetro	Descrição	Valor
n	número de elementos em cada cromossomo	quantidade total de itens
P	tamanho da população	50
P_e	tamanho da elite da população	10
α	valor do viés do cruzamento	0,7
número de gerações	quantidade de rodadas de cruzamentos e mutações realizadas	10

Tabela 1: Parâmetros BRKGA

Aqui estão presentes os resultados obtidos de um grupo de instâncias conhecidas na literatura, que podem ser obtidas pelo ESICUP (`https://www.euro-online.org/websites/esicup/data-sets/`). Elas variam em características como quantidade de itens e presença ou não de itens não-convexos.

Um detalhe importante é que como optamos por interpretar o caminho de corte como todo percurso que contém tanto as arestas de cada item como os movimentos ociosos realizados, não conseguimos comparar nossos resultados do caminho de corte e da soma ponderada com os da literatura (Oliveira [2019]) que abordam o mesmo problema integrado, porém considerando apenas os movimentos ociosos. Desse modo, comparamos apenas o valores obtidos quando otimizamos o comprimento de empacotamento.

Além disso, realizamos testes com 3 funções *fit* diferentes. Primeiramente, utilizamos o valor do comprimento de empacotamento, no teste seguinte utilizamos o valor do caminho de corte e, por fim, uma soma ponderada desses valores. Neste último caso, realizamos uma normalização dos valores do caminho de corte e do comprimento de empacotamento devido à disparidade entre suas grandezas e, após isso, multiplicamos cada um dos valores normalizados por 0,5 e os somamos para obter o valor da função fit_3. Para normalizar o caminho de corte, dividimos todos os valores calculados no terceiro teste pelo maior caminho de corte encontrado nos testes da primeira função *fit* e, analogamente, normalizamos o comprimento do empacotamento dividindo os valores pelo maior comprimento calculado nos testes da segunda função. Esse processo foi repetido e recalculado para cada instância. A ideia por trás disso era verificar se seria possível encontrar leiautes intermediários, que otimizassem ambos os valores mesmo sem atingir uma otimalidade em nenhum deles. As Tabelas 2, 3 e 4 apresentam os resultados obtidos pela aplicação do método implementado para resolver as instâncias selecionadas, bem como algumas características dessas instâncias. As colunas "Leiaute" se referem ao comprimento do leiaute e as colunas "CC" se referem ao comprimento do tamanho de corte obtidos.

Função fit_1

Instância	N. de itens	Altura do recipiente	Melhor resultado		Média dos resultados		T. médio (s)
			Leiaute	CC	Leiaute	CC	
Threep3w9	9	9	11,100	103,574	11,271	102,690	11,8
Fu7	7	38	24,000	267,048	24,000	267,048	4,8
Fu9	9	38	27,889	342,307	27,989	335,113	9,2
Fu10	10	38	30,000	372,398	32,314	387,482	11,9
Fu12	12	38	35,000	431,883	35,711	448,864	18,9
Shapes8	8	13	44,000	274,684	44,500	276,082	296,2
Jakobs1	25	25	19,000	369,093	19,175	376,893	979,8

Tabela 2: Resultado para otimização do comprimento de empacotamento

Os resultados obtidos se mostram promissores, uma vez que conseguimos alcançar bons resultados com uma população inicial pequena e pouca gerações. Para instâncias pequenas e/ou com prevalência de itens convexos, conseguimos isso em pouco tempo, como visto nos casos das instâncias Fu e Three. Para casos em que há um maior número de itens, o tempo computacional cresce drasticamente, porém ainda assim conseguimos bons resultados, como observado em Jakobs1. Para instâncias com poucos itens porém com muitos itens irregulares, também notamos um aumento no tempo, uma vez que na montagem do leiaute, as verificações por NFP para um polígono irregular acontecem em todos os polígonos que o compõe. Este é o caso de Shapes8.

Função fit_2

Instância	N. de itens	Altura do recipiente	Melhor resultado		Média dos resultados		T. médio (s)
			Leiaute	CC	Leiaute	CC	
Threep3w9	9	9	11,800	97,487	12,558	97,673	12,0
Fu7	7	38	28,000	242,213	27,600	247,931	4,0
Fu9	9	38	34,000	298,012	30,900	308,801	8,8
Fu10	10	38	38,000	342,782	36,457	348,556	11,4
Fu12	12	38	45,663	396,830	42,243	416,069	20,3
Shapes8	8	13	55,000	264,199	57,499	266,284	300,7
Jakobs1	25	25	21,000	369,289	21,408	372,969	975,7

Tabela 3: Resultado para otimização do caminho de corte

Função fit_3

Instância	Total de itens	Altura do recipiente	Melhor resultado		Média dos resultados		T. médio (s)
			Leiaute	CC	Leiaute	CC	
Threep3w9	9	9	11,400	99,154	11,450	99,788	11,8
Fu7	7	38	24,000	227,549	24,000	255,118	4,0
Fu9	9	38	29,000	300,233	29,000	312,392	8,1
Fu10	10	38	30,000	373,542	32,967	358,582	11,0
Fu12	12	38	35,143	446,283	36,955	437,020	19,7
Shapes8	8	13	44,000	274,684	44,500	275,039	316,4
Jakobs1	25	25	19,000	362,491	19,408	373,914	1030,6

Tabela 4: Resultado para otimização de ambos os problemas simultaneamente

Um resultado importante foi observado nos resultados da função fit_3. Em alguns casos, o método conseguiu alcançar resultados de caminho de corte superiores aos obtidos usando a função fit_2, que otimizava somente ele. Em outros, o método com esta função obteve resultados levemente inferiores, porém com grandes ganhos no leiaute. Isso pode ser justificado pelo fato de que leiautes com comprimento menor tendem a juntar mais os itens e sobrepor suas arestas. Isso permite menos movimentos ociosos e pode implicar um menor caminho de corte. Notou-se também que ele, ainda que não tenha produzido nenhum resultado com comprimento de leiaute menor àqueles obtidos pela função fit_1, na maioria dos casos obteve resultados muito próximos.

5. Considerações finais e trabalhos futuros

Diversos processos produtivos possuem a necessidade de cortar objetos maiores em peças menores de diversos formatos. O problema de empacotamento de itens irregulares em faixa tem como objetivo agrupar um conjunto de itens convexos e/ou não convexos em um recipiente de altura fixada e comprimento ilimitado, minimizando o comprimento total gasto pelas peças. Após definirmos um leiaute (disposição das peças no recipiente), naturalmente nos deparamos com a questão de qual é a melhor forma de cortar os itens, uma vez que este processo pode ser custoso, tanto em tempo, quanto em maquinário. Este é o problema de determinação do caminho de corte, que visa encontrar qual a sequência de movimentos a serem tomados por uma ou mais máquinas a fim de cortar todos os itens do leiaute.

A forma com que inserimos os itens e montamos o leiaute influencia diretamente nos resultados obtidos para o caminho mínimo de corte, como visto em Sherif et al. [2014]. Temos conhecimento de apenas 2 outros trabalhos que trataram o problema integrado: Anand e Babu [2015] que resolve apenas para peças retangulares e Oliveira et al. [2020], que utiliza uma discretização do recipiente e uma *math*eurística para resolver o problema integrado.

A priori, iniciamos a resolução do problema de empacotamento através do método heurístico *Bottom-left* que, dada uma sequência ordenada de itens, os posiciona um a um no recipiente, seguindo a sequência e sempre alocando cada item na posição mais à esquerda e abaixo possível. Após isso, transformamos o leiaute obtido em um grafo, de modo a unir as arestas e vértices sobrepostos. Com o grafo em mãos, utilizamos a modelagem do RPP para definir um modelo exato para o problema de determinação de caminho de corte e assim encontrar o caminho mínimo relativo ao grafo. Por fim, utilizamos o algoritmo genético BRKGA para gerar indivíduos que definem diferentes ordens de inserção e executam toda essa rotina para calcular os valores do caminho de corte e do comprimento do leiaute associados a cada um desses indivíduos. Com o passar das gerações, soluções cada vez melhores são encontradas.

Bons resultados foram obtidos pelo método proposto, ainda que com população reduzida e poucas gerações. Eles nos mostraram que abordagem por soma ponderada (função fit_3) é bastante promissora e alcança bons resultados quando comparados aos outras às outras funções *fit*. Além disso, foi possível observar que o tempo de execução é sensível à quantidade de itens, bem como a presença ou não de itens convexos.

Como não foram realizados numerosos testes, esta é uma importante direção de futuros trabalhos, uma vez que assim poderemos ter uma melhor compreensão do funcionamento do método, e com isso poderíamos realizar uma melhor calibração dos parâmetros do modelo. Os resultados da função fit_3 nos mostram que podemos obter bons casos otimizando ambos os parâmetros, e um estudo maior sobre a normalização empregada bem como sobre um possível ponderação dos valores pode nos levar a resultados ainda mais satisfatórios. Outro detalhe relevante seria implementar o método em uma linguagem de programação de maior eficiência, como o C++, para que possamos extrair ainda mais dele. Por fim, a adição de rotação dos itens poderia nos permitir resultados melhores e mais generalistas para aplicações práticas.

Agradecimentos

À Fundação de Amparo à Pesquisa do Estado de São Paulo - FAPESP (processos 2013/07375-0 e 2021/14949-9), pelo financiamento da pesquisa.

Referências

Álvarez-Valdés, R., Carravilla, M. A., e Oliveira, J. F. (2018). Cutting and packing. In *Handbook of Heuristics*, p. 1–46. Springer International Publishing.

Anand, K. V. e Babu, A. R. (2015). Heuristic and genetic approach for nesting of two-dimensional rectangular shaped parts with common cutting edge concept for laser cutting and profile blanking processes. *Computers & Industrial Engineering*, 80:111 – 124.

Bennell, J. e Oliveira, J. (2009). A tutorial in irregular shape packing problems. *Journal of the Operational Research Society*, 60:S93–S105.

Bennell, J. A. e Oliveira, J. F. (2008). The geometry of nesting problems: A tutorial. *European Journal of Operational Research*, 184(2):397 – 415.

Christofides, N., Campos, V., Corberán, A., e Mota, E. (1981). An algorithm for the rural postman problem.

Fowler, R. J., Paterson, M. S., e Tanimoto, S. L. (1981). Optimal packing and covering in the plane are NP-complete. *Information Processing Letters*, 12(3):133 – 137.

Lenstra, J. K. e Kan, A. H. G. R. (1976). *On general routing problems*. Networks, 6, pp. 273-280.

Leão, A. A., Toledo, F. M. B., Oliveira, J. F., Carravilla, M. A., e Alvarez-Valdés, R. (2020). Irregular packing problems: A review of mathematical models. *European Journal of Operational Research*, 282(3):803 – 822.

Manber, U. e Israni, S. (1984). Pierce point minimization and optimal torch path determination in flame cutting. *Journal of Manufacturing Systems*, 3(1):81 – 89.

Moreira, L. M., Oliveira, J. F., Gomes, A. M., e Ferreira, J. S. (2007). Heuristics for a dynamic rural postman problem. *Computers & Operations Research*, 34(11):3281 – 3294.

Mundim, L. R., Andretta, M., e Queiroz, T. A. (2017). A biased random key genetic algorithm for open dimension nesting problems using no-fit raster. *Expert Systems with Applications*, 81: 358–371.

Oliveira, L. T. (2019). *Uma integração dos problemas de empacotamento de peças irregulares e do caminho mínimo de corte*. PhD thesis, Universidade de São Paulo.

Oliveira, L. T., Silva, E. F., Oliveira, J. F., e Toledo, F. M. B. (2020). Integrating irregular strip packing and cutting path determination problems: A discrete exact approach. *Computers & Industrial Engineering*, 149:106757.

Sherif, S. U., Jawahar, N., e Balamurali, M. (2014). Sequential optimization approach for nesting and cutting sequence in laser cutting. *Journal of Manufacturing Systems*, 33(4):624 – 638.

Silva, E. F., Oliveira, L. T., Oliveira, J. F., e Toledo, F. M. B. (2019). *Exact approaches for the cutting path determination problem*. Computers and Operations Research 112.

Silva, E. F., Oliveira, L. T., e Toledo, F. M. B. (2015). Uma nova abordagem para o problema de determinação do caminho de corte. *XLVII Simpósio Brasileiro de Pesquisa Operacional*, p. 4087 – 4097.

Usberti, F. L., França, P. M., e França, A. L. M. (2011). The open capacitated arc routing problem. *Computers & Operations Research*, 38(11):1543 – 1555.

Wäscher, G., Haußner, H., e Schumann, H. (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109 – 1130.