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

MÉTODOS DE SOLUÇÃO PARA UM PROBLEMA DE CORTE DE ITENS IRREGULARES APLICADO À INDÚSTRIA SIDERÚRGICA

Sarah Lopes Lira Feitosa, Walison Adrian de Oliveira e Marina Andretta
Universidade de São Paulo
Av. Trabalhador São Carlense, 400 - Centro, São Carlos - SP
sarah.mat020@usp.br, walisonadrian201@usp.br,
andretta@icmc.usp.br

RESUMO

Uma empresa da área siderúrgica necessita cortar placas de aço em itens diversos, procurando atender ao pedido do cliente e utilizando a menor quantidade possível de placas, a fim de diminuir custos e descarte de material. Os itens são irregulares e podem rotacionar livremente. Tal problema é chamado de corte de itens irregulares, que consiste em arranjar itens irregulares, convexos ou não convexos, em qualquer rotação, em recipientes retangulares. Por ser um problema NP-difícil, encontrar uma solução ótima pode ser muito custoso, o que nos fez optar por encontrar uma solução aproximada. Usando como inspiração essa aplicação, um problema foi resolvido utilizando uma metaheurística evolutiva chamada *Biased Random-Key Genetic Algorithm*, que utiliza de uma heurística construtiva chamada *Bottom-left*. O método de solução utilizado apresentou bons resultados para as instâncias adaptadas da literatura, se mostrando promissor para ser usado para resolver problemas reais da indústria siderúrgica.

PALAVRAS CHAVE. Problema bidimensional de corte de itens irregulares, Metaheurísticas, Indústria.

Tópicos: POI - PO na Indústria, MH - Metaheurística

ABSTRACT

A company in the steel industry needs to cut steel plates into different items, trying to meet the customer's request and using the smallest possible amount of plates, in order to reduce costs and material disposal. Items are irregular and can rotate freely. Such a problem is called irregular item cutting, which consists of arranging irregular items, convex or non-convex, in any rotation, in rectangular containers. As it is an NP-hard problem, finding an optimal solution can be very costly, which made us choose to find an approximate solution. Using this application as inspiration, the problem was solved using an evolutionary metaheuristic called Biased Random-Key Genetic Algorithm, that uses a constructive heuristic called Bottom-left. The solution method used showed good results for the instances adapted from the literature, showing it can be used to solve real problems in the steel industry.

KEYWORDS. Two-dimensional Irregular Cutting Problem, Metaheuristics, Industry.

Topics: POI - PO in Industry, MH - Metaheuristics

https://proceedings.science/p/175165?lang=pt-br

1. Introdução

A otimização do processo de corte de materiais desempenha um papel fundamental na eficiência e redução de custos em diversos setores industriais. Na indústria siderúrgica, em particular, a maximização do aproveitamento do aço e a minimização do desperdício são importantes para aumentar a competitividade e sustentabilidade das empresas. Neste contexto, o problema de corte de itens irregulares surge para atender essa demanda do setor industrial siderúrgico, pois envolve a alocação eficiente de itens com formas e tamanhos variados em uma placa retangular, como chapas de aço, visando minimizar o desperdício da placa. Por serem irregulares, a alocação desses itens pode ser um pouco mais complexa, tornando a busca por soluções ótimas algo muito exigente computacionalmente.

Nesse artigo, exploramos um método de solução para o problema de corte de itens irregulares semelhante ao problema da indústria siderúrgica, o método metaheurístico BRKGA *(Biased Random-Key Genetic Algorithm)*, que utiliza a heurística do *Bottom-left* para encontrar soluções. Por se tratar de um problema de difícil solução e por estarmos interessados em resolver instâncias maiores do problema, optamos por essa abordagem heurística no lugar de uma abordagem exata.

A metaheurística BRKGA, proposta por Gonçalves e Resende [2011], é um algoritmo genético, que utiliza uma abordagem baseada na evolução biológica para buscar melhores soluções aproximadas do problema. Já a heurística construtiva *Bottom-left* é empregada para encontrar uma solução em si, visando otimizar a utilização da placa disponível.

O objetivo deste trabalho é comparar a eficiência das soluções obtidas pelo método, considerando como critério o aproveitamento do espaço da placa e o tempo de resolução, para que possamos avaliar a viabilidade desse método ser utilizado na indústria siderúrgica, uma vez que esse setor industrial é muito antigo e, em alguns casos, pouco desenvolvido tecnologicamente para o corte otimizado de placas de aço.

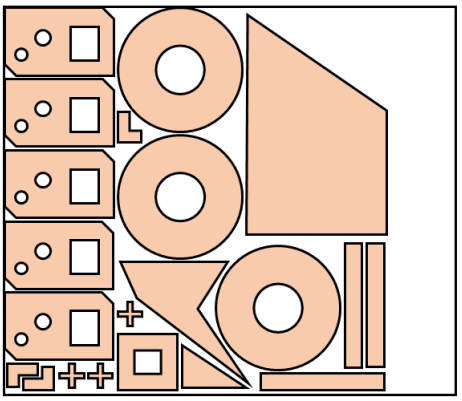
Esperamos que os resultados deste trabalho contribuam para a compreensão e o avanço dos métodos de solução para o problema de corte de itens irregulares, principalmente para a indústria siderúrgica e abrindo caminho para novas pesquisas na área.

Este trabalho está organizando da seguinte forma. Na Seção 2, temos a definição do problema que buscamos resolver. Na Seção 3, apresentamos alguns métodos de solução presentes na literatura para resolver problemas similares e o método proposto para o problema definido na Seção 2. Na Seção 4 apresentamos os resultados obtidos na aplicação do método proposto para resolver algumas instâncias da literatura. Na Seção 5 apresentamos as considerações finais e propostas de trabalhos futuros.

2. Definição do problema

Várias indústrias, como as de papel, têxtil e siderúrgica, buscam cortar seus materiais (papel, tecido e aço) da melhor forma possível, ou seja, procuram diminuir o desperdício de material. No nosso caso, uma empresa do estado de São Paulo da área siderúrgica necessita cortar placas de aço em itens diversos, dependendo do pedido de um dado cliente. Os itens podem ter qualquer tipo de formato, inclusive podem ter buracos, e podem ser rotacionados livremente. As placas são retangulares de tamanho e espessuras fixos e, ao fazer o corte, a perda de material (chamada de sangria) é proporcional à espessura da placa. O objetivo é fazer o corte dos itens nas placas de modo a otimizar o espaço, para que, posteriormente, possa se utilizar o que sobrou da placa para novos pedidos, trazendo sustentabilidade à empresa e diminuição de custos. A Figura 1 mostra um exemplo de como é feita a distribuição dos itens em uma placa de aço.

Figura 1: Ilustração de um exemplo de corte de itens em uma placa de aço.



Assim, seu objetivo é cortar as placas de aço de tamanho fixo, procurando atender ao pedido do cliente e utilizando o mínimo possível de material.

Pela complexidade de se resolver esse problema, o simplificamos considerando que os itens serão apenas polígonos convexos e não convexos. Cada polígono não convexo é particionado e representado como um conjunto de polígonos convexos. Além disso, cada item tem uma quantidade de cópias a serem alocadas e rotações fixas dadas. Supomos que os polígonos representam o item a ser cortado, já considerando a sangria.

Como possuem uma espessura predefinida, cada item pode ser cortado apenas de um tipo de placa. Por isso, sem perda de generalidade, podemos supor que todos os itens a serem cortados tem a mesma espessura e todas as placas são iguais (caso haja itens de espessuras diferentes, basta resolver vários subproblemas, cada um com itens e placas de espessuras iguais).

Consideramos que temos um conjunto de placas (recipientes) retangulares iguais, com largura e altura fixos, e o objetivo é minimizar o espaço utilizado desses recipientes, fazendo o corte de todos os itens, sem sobreposição entre si. Assim, definimos nossa função objetivo como minimizar a quantidade de recipientes usados para cortar todos os itens.

Este problema é conhecido na literatura como *Single Stock Size Cutting Stock Problem* (SSSCSP) chamado de problema de corte de tamanho de estoque único que pode ser encontrado em Wäscher et al. [2007]. No SSSCSP, temos um conjunto dado de m itens, cada um com sua quantidade de cópias, e um estoque de recipientes retangulares de largura W e altura H fixos. O objetivo é minimizar a quantidade de recipientes utilizados para cortar todos os itens.

Para nossa implementação, cada item i é representado por um conjunto P_i de polígonos convexos, e possui, além de uma quantidade de cópia, um conjunto finito R_i de possíveis rotações. Consideramos que os recipientes retangulares estão no plano cartesiano, com seu canto inferior esquerdo posicionado em $(0, 0)$.

O SSSCSP é um problema NP-difícil, então não se conhece um algoritmo em tempo polinomial que o resolva. Por isso, optamos por resolvê-lo usando uma metaheurística. Na seção a seguir, faremos uma pequena revisão da literatura sobre métodos para resolver problemas similares a este e, em seguida, apresentamos o método escolhido para resolvê-lo.

3. Método de Solução

Vários trabalhos resolveram problemas semelhantes ao SSSCSP, com itens irregulares que permitem ser rotacionados para serem cortados ou alocados em um ou mais recipientes. Em Peralta et al. [2018] e Cherri et al. [2018], são resolvidos problemas de empacotamento de itens irregulares

a partir de métodos exatos em que é permitida a rotação livre dos itens. Por outro lado, temos em Abeysooriya et al. [2017], Martinez-Sykora et al. [2016] e Mundim et al. [2017] abordagens heurísticas para resolver um problema de empacotamento de itens irregulares com rotação livre, a partir do algoritmo *Jostle*, de uma *math*-heurística, e de uma metaheurística, respectivamente.

Embora os métodos exatos sejam capazes de encontrar uma melhor solução, essa busca pode demandar muito tempo e memória computacional quando lidamos com problemas NP-difíceis. Por isso, para resolver o SSSCSP, optamos por implementar a metaheurística *Biased Random-Key Genetic Algorithms* (BRKGA), proposta por Gonçalves e Resende [2011]. Esta metaheurística já foi utilizada com sucesso em outros trabalhos para resolver problemas de corte de itens irregulares Mundim et al. [2017]; Pinheiro et al. [2016]. Para criar o decodificador exigido pelo BRKGA, utilizamos a heurística *Bottom-left*, uma das heurísticas construtivas mais usadas para resolver problemas de corte irregular Baker et al. [1980]. A seguir, detalhamos cada um destes métodos.

3.1. Heurística *Bottom-left*

Como mencionado anteriormente, para construir uma solução do problema SSSCSP, utilizamos uma heurística *Bottom-left* Baker et al. [1980], que é uma das mais usadas (em diferentes variações) para encontrar soluções para problemas de corte de itens irregulares.

Desenvolvida para resolver problemas em que o objetivo é minimizar a largura utilizada do recipiente para alocar todos os itens, a ideia básica desta heurística é, dada uma lista de itens, em uma certa ordem, inserir os itens no recipiente um a um, sempre na posição mais à esquerda e abaixo possível. No nosso caso, dada uma ordenação para todas as cópias de todos os itens, cada uma com uma rotação definida, tentamos inserir cada item no primeiro recipiente o mais à esquerda e abaixo possível. Se um item não cabe no recipiente, ele é pulado e passamos para o próximo. Depois de percorrer a lista toda de itens e inserir o que foi possível no primeiro recipiente, passamos para o segundo recipiente e repetimos o processo com os itens ainda não inseridos. Repetimos estes passos até que todos os itens sejam inseridos em algum recipiente.

Para inserir os itens no recipiente, precisamos garantir que todos eles fiquem inteiramente contidos no recipiente e não haja sobreposições. Neste trabalho, cada item possui um de seus vértices como ponto de referência, e este ponto é usado para alocar o item no recipiente. Para garantir a contenção dos itens, usamos o conceito de IFP (*Inner-Fit Polygon*), que consiste no lugar geométrico dos pontos do recipiente em que um dado item pode ter seu ponto de referência alocado, de forma que o item fique inteiramente contido no recipiente. O IFP é facilmente calculado, mas é diferente para cada rotação do item.

Para evitar a sobreposição entre itens, usamos o conceito de NFP (*No-Fit Polygon*). Para cada par de itens i e j, com o item i já alocado no recipiente, o NFP_{ij} é um polígono tal que, se o ponto de referência de j está na borda do NFP_{ij}, o item j está encostado no item i; se o ponto de referência de j está no interior de NFP_{ij}, o item j está sobreposto ao item i; se o ponto de referência de j está fora do NFP_{ij}, o item j está separado do item i. O cálculo do NFP é custoso e, por isso, é feito em uma etapa de pré-processamento. Vale também ressaltar que, em geral, quando um dos itens é rotacionado, o NFP correspondente muda. Por isso, precisamos construir um NFP diferente para cada par de itens i e j, em cada uma de suas rotações permitidas. Para mais detalhes sobre IFP e NFP, veja Bennell e Oliveira [2008].

Como sempre que vamos inserir um item ele tocará a borda do recipiente e/ou de outros itens, podemos usar os próprios IFPs e NFPs para determinar o ponto mais à esquerda e abaixo em que o ponto de referência de um item deve ser inserido.

O Algoritmo 1 sintetiza os passos da heurística *Bottom-left* utilizada.

Algorithm 1 *Bottom-left* utilizando NFP e IFP para o SSSCSP

Require: Lista $I = (i_1, \ldots, i_n)$ de n itens na ordem de inserção, cada um na rotação em que será empacotado; lista $R = (r_1, \ldots, r_n)$ de recipientes vazios, com largura W e altura H

Ensure: Lista S com os recipientes usados e itens posicionados em cada um deles; a quantidade de recipientes utilizados $n_p = |S|$; a menor largura l_{min} necessária dentre as placas para posicionar os itens

Defina $n_p = 0$ e $l_{min} = W$
while lista I não for vazia **do**
    Selecione r o primeiro recipiente da lista R
    for cada elemento i da lista I **do**
        Calcule, com base no IFP e NFP e i com todos os itens já empacotados em r, o ponto p mais à esquerda e abaixo de r em que i pode ser inserido
        if foi possível calcular p **then**
            Insira i na posição p de r
            Remova i da lista L
        end if
    end for
    Remova r da lista R e o insira na lista S
    Some um ao valor de n_p
    Atualize l_{min} considerando o valor da largura usada de r
end while

3.2. *Biased Random-Key Genetic Algorithm* (BRKGA)

O BRKGA, proposto em Toso e Resende [2015], é uma metaheurística bio-inspirada baseada no Algoritmo Genético (GA) de Holland [1975] para encontrar uma boa solução. Por ser uma heurística evolutiva, a cada iteração, busca-se melhorar cada vez mais as soluções a partir de perturbações nas soluções obtidas anteriormente. Essa evolução é baseada em uma analogia com a teoria da evolução das espécies iniciada pelo inglês Charles Darwin. Utilizaremos uma versão do BRKGA baseada na proposta em Mundim et al. [2017].

Para a resolução do SSSCSP, a analogia é feita da seguinte forma:

- Tem-se uma população P de indivíduos (soluções) de uma mesma espécie em que cada indivíduo é composto por um conjunto de n cromossomos (ordem para alocar n itens).
- Cada indivíduo possui um *fitness* (um valor que representa a solução) tal que, quanto melhor for esse *fitness*, maiores as chances daquele indivíduo (solução) ser o mais apto a sobreviver da sua população. Dessa forma, os indivíduos da população são classificados em populações elite P_e e não-elite P_{ne}, a partir dos valores de *fitness* de cada um.
- Há, então, a necessidade de formar uma nova geração para que a espécie perpetue. Essa necessidade pode ser satisfeita por: mutação, em que se muda a posição de um ou mais cromossomos de indivíduos, dando origem a novos, formando a população de mutantes P_m; ou por cruzamento de dois indivíduos, que forma um novo indivíduo a partir da combinação de cromossomos dos dois indivíduos, dando origem à população P_c.
- Em seguida, os indivíduos da geração que possuem melhores *fitness* permanecem na nova geração, enquanto os outros são descartados para que novos filhos sejam gerados sem au-

https://proceedings.science/p/175165?lang=pt-br

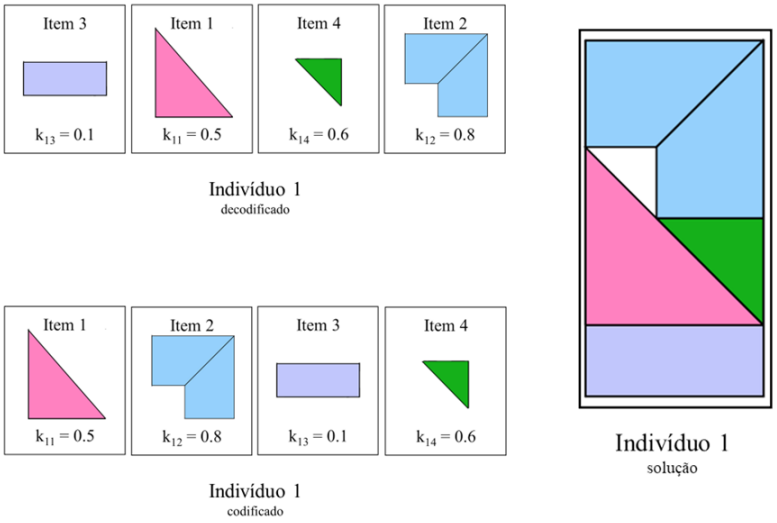
mento da população. Repete-se o processo até que o critério de parada do algoritmo seja satisfeito.

Detalhamos a seguir cada aspecto do BRKGA e como cada indivíduo de cada população são gerados.

- **População Inicial.** Conjunto de indivíduos em que cada indivíduo i possui um conjunto de n cromossomos que vão representar os n itens, cada um com uma dada rotação, que serão alocados no recipiente. A ordem que cada cromossomo aparece em um dado indivíduo é a ordem codificada a partir de uma chave que será explicada a seguir.

- **Decodificação de um indivíduo.** Para cada cromossomo j de um indivíduo i, teremos uma chave $k_{ij} \in [0, 1]$ que será utilizada para decodificar a ordem em que os itens serão alocados. A decodificação é feita ordenando os itens seguindo a ordem crescente das chaves aleatórias a eles associadas. A partir dessa ordem, a decodificação para encontrar o valor do *fitness* é feita pela heurística *Bottom-Left*, explicada na Seção 3.1. Na Figura 2 temos um exemplo de um indivíduo i, as chaves k_{ij} de cada cromossomo j e sua decodificação.

Figura 2: Ilustração de um exemplo de decodificação de um indivíduo



- ***Fitness*.** É um valor que classifica cada indivíduo como uma boa solução ou não em relação aos outros indivíduos. Em Mundim et al. [2017], o valor do *fitness* para o problema de duas dimensões aberto é a área do menor retângulo que envolve os itens. No nosso caso, o *fitness* é a quantidade de placas que são utilizadas para alocar todos os itens (n_p). Assim, indivíduos com menor *fitness* são melhores soluções. Caso exista mais de um indivíduo com a mesma quantidade de placas usadas na sua solução, o critério de desempate é a menor largura utilizada entre as placas ocupadas (l_{min}). O *fitness* foi escolhido dessa forma, pois a intenção é utilizarmos a menor quantidade de placas para fazer o corte dos itens, mas sabendo que

placas podem ser utilizadas para outros pedidos, diminuindo custos e a quantidade de sucata produzida pela empresa. Não consideramos o reaproveitamento de placas já usadas neste trabalho, mas o método aqui proposto pode ser facilmente adaptado para considerar este caso, bastando considerar que itens já cortados estão alocados nas respectivas placas e não podem ter suas posições mudadas.

- **Populações elite e não-elite.** A população total P de uma dada geração é dividida em duas populações a partir do *fitness* de cada indivíduo: P_e e P_{ne}. Assim como em Mundim et al. [2017], a população elite P_e será 20% da população dada dos indivíduos que possuem melhores *fitness*, enquanto a população não-elite P_{ne} será os 80% restantes de P. Os indivíduos de P_e geralmente são os mais aptos a terem filhos de melhores soluções para as novas gerações. Posteriormente, os indivíduos de P_{ne} serão descartados e não participarão da próxima geração.

- **Mutação.** Para geração de novos filhos, realiza-se uma mutação com uma taxa de 10% da população da geração, como em Mundim et al. [2017]. No entanto, em vez de gerar indivíduos mutantes de forma aleatória (como em Mundim et al. [2017]), os indivíduos mutantes são gerados da seguinte forma: é escolhido aleatoriamente um indivíduo em P_e e escolhe-se (também aleatoriamente) dois cromossomos desse indivíduo para serem permutados. Essa mudança na ordem cromossômica gera um novo indivíduo que fará parte da população de mutantes P_m que será adicionada à nova população P_{nova}.

- ***Biased Parametrized Uniform Crossover*.** Ainda para gerar novos filhos, tem-se o processo de cruzamento (*crossover*), em que escolhemos dois indivíduos, um de P_e e outro de P_{ne}, e os combinamos para formar um novo indivíduo. O cruzamento utilizado nesse algoritmo é o *Biased Parametrized Uniform Crossover*, como em Mundim et al. [2017]. Ele funciona da seguinte forma: escolhemos aleatoriamente um indivíduo de P_e e outro de P_{ne} e, com uma probabilidade ρ, escolhemos de qual dos indivíduos escolheremos um cromossomo para fazer parte do novo indíviduo que fará parte da população P_c (que é 70% do tamanho de P). Dessa forma, teremos novos indivíduos que serão uma combinação de indivíduos de melhor e pior *fitness*, dando iguais chances de se escolher um cromossomo deles, para termos maior variabilidade genética. O valores de ρ usados por Mundim et al. [2017] foram 0.5 e 0.7. Aqui utilizamos apenas $\rho = 0.5$.

- **Nova população.** A população P_{nova} será formada por P_e, P_m e P_c. Todos os indivíduos que foram formados nessa geração serão decodificados e serão classificados novamente para que outra geração possa ser formada, visando sempre a evolução das gerações para se encontrar soluções cada vez melhores para o problema.

A seguir temos o Algoritmo 2, que sintetiza como o BRKGA é usado.

Algorithm 2 BRKGA para o SSSCSP

Require: População P inicial de indivíduos codificados; tamanho da população t_{pop}; quantidade máxima de gerações g_{max}; probabilidade ρ utilizada no *Biased Parametrized Uniform Crossover*

Ensure: Solução x^* de melhor *fitness* encontrada

Defina $g = 1$
while $g \leq g_{max}$ **do**
    Decodifique P e encontre o *fitness* de cada indivíduo a partir da solução encontrada pela heurística *Bottom-left* descrita no Algoritmo 1
    Divida P em dois grupos: P_e com 20% de P de indivíduos com melhores *fitness* e o restante em P_{ne}
    Forme um novo grupo P_m a partir de 10% dos indivíduos de P_e, permutando dois cromossomos aleatórios do indivíduo escolhido
    Adicione a uma nova população P_{nova} os indivíduos dos grupos P_e e P_m
    Forme um novo grupo P_c de tamanho $|P - P_m - P_e|$ de indivíduos a partir do *Biased Parametrized Uniform Crossover* com uma probabilidade ρ
    Adicione a P_{nova} os novos indivíduos gerados de P_c
    Encontre o indivíduo x_{nova} de melhor *fitness* de P_{nova}
    if $fitness(x_{nova}) < fitness(x^*)$ **then**
        Atualize x^* com x_{nova}
    end if
    Some um ao valor de g
end while

4. Experimentos numéricos

Implementamos a metaheurística descrita no Algoritmo 2 em linguagem Python 3.8.8. Escolhemos o Jupyter Notebook (`https://jupyter.org/`) como ambiente de execução para o algoritmo. Todos os experimentos foram realizados em um computador com processador Intel Core™ i7-2600 de 3.40 GHz, com 16 GB de memória RAM e sistema operacional Ubuntu 20.04.4 LTS.

Pela linguagem de programação escolhida, em experimentos preliminares verificamos que os tempos de execução do método implementado podem ser relativamente altos. Por isso optamos por usar o tamanho da população do BRKGA como 30 e um máximo de 5 gerações. Para cada instância, a metaheurística foi repetida 5 vezes devido à aleatoriedade que o método possui e reportamos a melhor solução encontrada, a pior e a solução média, bem como a média do tempo de execução (em segundos).

Para verificar a eficácia do método implementado para encontrar soluções boas para o problema em questão, foram utilizadas algumas instâncias da coleção presente no ESICUP (`https://www.euro-online.org/websites/esicup/data-sets/`), com 6 a 30 itens para serem alocados. A quantidade de cópias de cada item e suas respectivas rotações fixadas fazem parte das instâncias originais. Os NFPs de cada combinação dos itens e rotações foram calculados em uma fase de pré-processamento. Os recipientes possuem uma das dimensões H fixa fornecida por cada instância. Para escolher a outra dimensão W, fizemos de duas formas. Seja $A = HW$ a área da placa. Assim, temos:

1. $A = S$, com S sendo a soma das áreas de todos os itens a serem alocados (inclusive suas cópias). Assim, definimos uma dimensão $a = S/H$. Como podem existir instâncias em que

https://proceedings.science/p/175165?lang=pt-br

Tabela 1: Detalhes das instâncias utilizadas

Instância	Total de Itens	Possíveis rotações dos itens	Dimensões da placa $(W \times H)$
Threep2w9-1	6	0	7.7×9
Threep2w9-2	6	0	15.3×9
Dighe2-1	10	0	99.9×100
Dighe2-2	10	0	199.8×100
Fu-1	12	0, 90	28.5×38
Fu-2	12	0, 90	57×38
Poly1A-1	15	0, 90, 180, 270	13×40
Poly1A-2	15	0, 90, 180, 270	20.6×40
Dighe1-1	16	0	100×100
Dighe1-2	16	0	200×100
Blaze2-1	16	0, 180	15.1×15
Blaze2-2	16	0, 180	30.1×15
Jakobs1-1	25	0, 90, 180, 270	9.8×40
Jakobs1-2	25	0, 90, 180, 270	19.6×40
Blaze1-1	28	0, 180	21.6×15
Blaze1-2	28	0, 180	43.2×15
Poly2A-1	30	0, 90, 180, 270	20.6×40
Poly2A-2	30	0, 90, 180, 270	41.2×40
Dagli-1	30	0, 180	50.7×60
Dagli-2	30	0, 180	101.4×60

um ou mais itens podem não caber nessa dimensão em uma rotação específica, definimos também uma dimensão b como sendo a maior largura (dentre todas as rotações) dentre o conjunto de itens a serem alocados. Assim, definimos que $W = \max(a, b)$;

2. $A = 2S$, com S sendo a soma das áreas de todos os itens a serem alocados (inclusive suas cópias). Logo, temos que $W = 2S/H$.

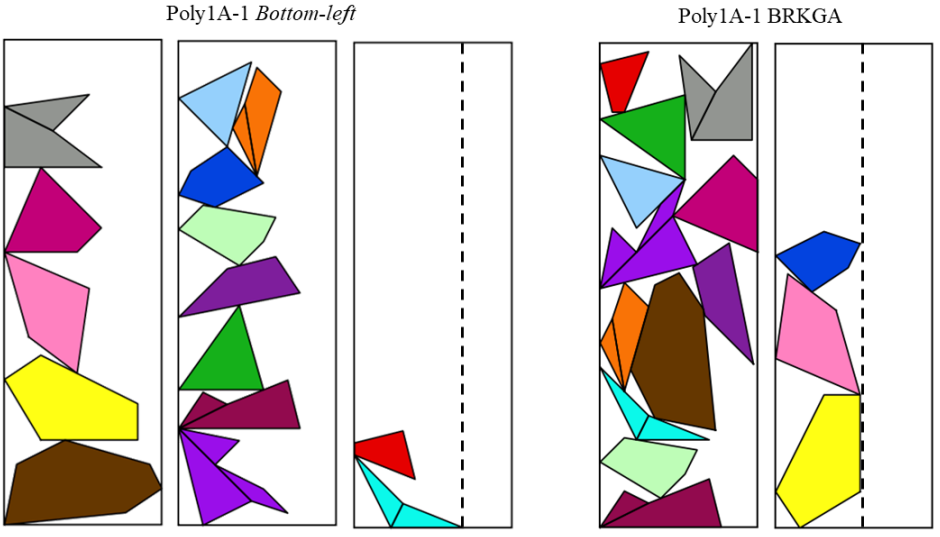
A Tabela 1 mostra as características das instâncias usadas. A Tabela 2 apresenta as soluções obtidas para cada instância utilizando tanto o BRKGA, como a heurística *Bottom-left* em que a ordem é feita de forma decrescente em relação à área dos itens, todos com a rotação original. Para a heurística *Bottom-left*, reportamos a solução encontrada (número de placas usadas n_p e largura mínima entre as placas l_{min}) e o tempo gasto, em segundos. Para o BRKGA, reportamos a melhor, pior e média das soluções encontradas em suas 5 execuções, bem como o tempo médio de execução (em segundos).

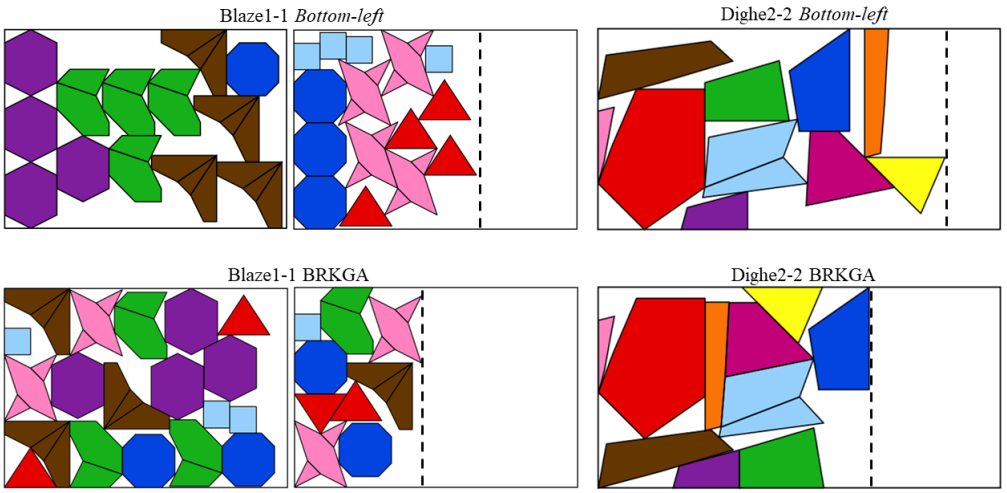
Embora o tempo para encontrar uma solução com o BRKGA seja muito maior que usando o *Bottom-left*, ao comparar os resultados, para a maioria das instâncias, a metaheurística encontrou soluções melhores que a heurística construtiva. Para três instâncias (Dighe2-1, Poly1A-1 e Poly1A-2), o BRKGA foi capaz de encontrar uma solução com uma placa a menos que a heurística *Bottom-left*. Para as 17 instâncias restantes, o BRKGA encontrou soluções com a mesma quantidade de placas, porém com menor largura (considerando sua pior solução) em 16 delas. Apenas no caso da instância Jakobs1-2 ambos métodos encontraram uma solução de qualidade similar. A Figura 3 ilustra as soluções encontradas pelos métodos para as instâncias Poly1A-1, Dighe2-2 e Blaze1-1.

Tabela 2: Resultados para as instâncias da ESICUP - recipiente original

		Bottom-left		BRKGA			
Instância	Total de Itens	Solução (n_p, l_{min})	Tempo (s)	Melhor solução (n_p, l_{min})	Média solução (n_p, l_{min})	Pior solução (n_p, l_{min})	Tempo médio (s)
Threep2w9-1	6	(2, 6.67)	0.02	(2, 6)	(2, 6)	(2, 6)	4.85
Threep2w9-2	6	(1, 12)	0.05	(1, 11.3)	(1, 11.4)	(1, 11.5)	12.07
Dighe2-1	10	(3, 40)	0.06	(2, 52)	(2, 54.14)	(2, 62.71)	13.1
Dighe2-2	10	(1, 172.4)	0.13	(1, 134.55)	(1, 138)	(1, 142.43)	34.19
Fu-1	12	(2, 15.2)	0.13	(2, 12)	(2, 13.16)	(2, 14)	10.63
Fu-2	12	(1, 42)	0.18	(1, 36)	(1, 36.56)	(1, 37.41)	23.91
Poly1A-1	15	(3, 9)	1.13	(2, 7.67)	(2, 8.03)	(2, 9)	103.2
Poly1A-2	15	(2, 9)	1.87	(1, 16.56)	(1, 17.3)	(1, 17.97)	302.71
Dighe1-1	16	(2, 79.06)	0.21	(2, 48.92)	(2, 58.24)	(2, 67.58)	39.27
Dighe1-2	16	(1, 175.82)	0.38	(1, 144.15)	(1, 146.53)	(1, 149.4)	115.55
Blaze2-1	16	(2, 13.5)	1.16	(2, 7.25)	(2, 7.43)	(2, 7.6)	388.84
Blaze2-2	16	(1, 26.17)	5.46	(1, 21.2)	(1, 21.51)	(1, 21.8)	1189.92
Jakobs1-1	25	(2, 8)	3.46	(2, 6)	(2, 6)	(2, 6)	344.29
Jakobs1-2	25	(1, 13)	5.1	(1, 13)	(1, 13.27)	(1, 13.69)	1196.4
Blaze1-1	28	(2, 13.9)	6.08	(2, 9.67)	(2, 10.18)	(2, 10.7)	2184.3
Blaze1-2	28	(1, 32.18)	24.79	(1, 30)	(1, 30.14)	(1, 30.43)	6999.6
Poly2A-1	30	(2, 20.08)	5.31	(2, 13.9)	(2, 15.5)	(2, 16)	871.94
Poly2A-2	30	(1, 36.11)	13.3	(1, 31.75)	(1, 32.37)	(1, 33.35)	3669.43
Dagli-1	30	(2, 34.19)	24.88	(2, 21.36)	(2, 22.31)	(2, 23.1)	2958.46
Dagli-2	30	(1, 73.12)	39.89	(1, 67.41)	(1, 67.94)	(1, 68.89)	8692.74

Figura 3: Ilustração de soluções obtidas para as instâncias Poly1A-1, Blaze1-1 e Dighe2-2 pelos métodos propostos





5. Considerações finais e trabalhos futuros

Neste trabalho, realizamos uma investigação sobre métodos de solução para o SSSCSP, com foco na sua aplicação na indústria siderúrgica. Nossa análise fez uso de um método metaheurístico BRKGA, que utilizou a heurística do *Bottom-left* para encontrar soluções.

O BRKGA mostrou-se interessante para encontrar soluções com boa qualidade para as instâncias resolvidas quando comparado ao uso apenas da heurística *Bottom-left*, apesar de ter um tempo de execução muito mais elevado. Das 20 instâncias testadas, em apenas 1 caso ambos os métodos obtiveram solução com qualidade equivalente. Nas 19 restantes, o BRKGA encontrou uma solução que utilizou menor quantidade de placas (3 instâncias) ou menor largura utilizada. Isso aponta que este método é promissor para resolver este problema.

Como trabalhos futuros, pretendemos realizar mais experimentos numéricos, com um número maior e mais variado de instâncias, a fim de identificar os pontos fortes e fracos do método. Faremos experimentos também com instâncias baseadas em casos reais, para analisar o quanto o método proposto é adequado para uso na indústria siderúrgica. Procuraremos também adaptar o método para permitir o uso, como recipientes, outras placas que já foram utilizadas para empacotar outras instâncias, acrescentando uma lista de placas já usadas para que sejam reaproveitadas, a fim de trazer menos desperdício de material à indústria e, consequentemente, maior sustentabilidade.

Além disso, pretendemos aperfeiçoar a implementação do método, usando estruturas de dados mais eficientes e uma linguagem como C/C++, pois isso fará com que o tempo de execução do BRKGA diminua significativamente.

Agradecimentos

Ao Programa Unificado de Bolsas (PUB) da Universidade de São Paulo e à FAPESP (processos 2013/07375-0 e 2022/08538-9), pelo financiamento da pesquisa, e ao aluno Leonardo Trevisan do ICMC-USP, por disponibilizar o código de uma versão do *Bottom-left* utilizada neste trabalho.

Referências

Abeysooriya, R., Bennell, J., e Martinez-Sykora, A. (2017). Jostle heuristics for the 2d-irregular shapes bin packing problems with free rotation. *International Journal of Production Economics*, 195.

Baker, B., Coffman, E., e Rivest, R. (1980). Orthogonal packings in two dimensions. *SIAM J. Comput.*, 9:846–855.

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

Cherri, L. H., Cherri, A., e Soler, E. (2018). Mixed integer quadratically-constrained programming model to solve the irregular strip packing problem with continuous rotations. *Journal of Global Optimization*, 72:1–19.

Gonçalves, J. e Resende, M. (2011). Biased random-key genetic algorithms for combinatorial optimization. *J. Heuristics*, 17:487–525.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI. second edition, 1992.

Martinez-Sykora, A., Alvarez-Valdes, R., Bennell, J., Ruiz, R., e Tamarit, J. (2016). Matheuristics for the irregular bin packing problem with free rotations. *European Journal of Operational Research*, 258.

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

Peralta, J., Andretta, M., e Oliveira, J. (2018). Solving irregular strip packing problems with free rotations using separation lines. *Pesquisa Operacional*, 38.

Pinheiro, P. R., Júnior, B. A., e Saraiva, R. D. (2016). A random-key genetic algorithm for solving the nesting problem. *International Journal of Computer Integrated Manufacturing*, 29(11):1159–1165.

Toso, R. e Resende, M. (2015). A c++application programming interface for biased random-key genetic algorithms. *Optimization Methods and Software*, 30.

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.