Genetic Algorithm for the Knapsack Problem with Irregular Shaped Items

1st Layane R. S. Queiroz
3rd Marina Andretta
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo
São Carlos-SP, Brasil
layaner@usp.br
andretta@icmc.usp.br

2nd Leandro R. Mundim
Optimized Decision Making
São Carlos-SP, Brasil
mundim@icmc.usp.br

***Abstract*—The two-dimensional knapsack problem with irregularly shaped items is solved in this work. It is utilized the concept of inner-fit raster and no-fit raster to verify packing feasibility, which stands for non-overlapping between items that are entirely contained inside the bin. The problem solution is obtained with a biased random-key genetic algorithm in which each chromosome contains information related to the order and rotation where each item should be packed into the bin. The chromosome also contains information about which heuristic has to be used to pack items and the probability of an offspring inheriting information from an elite parent. It is adopted three heuristics for positioning items, which are: bottom-left, left-bottom, and horizontal zig-zag. The experiments over literature instances showed that the developed genetic algorithm is very effective since it could obtain an optimal solution for 53.4% of the instances and improved the bin's occupancy ratio in about 2.1% when observing all the instances.**

***Index Terms*—Biased Random-Key Genetic Algorithm; Two-dimensional Knapsack Problem; Irregularly shaped items; Nesting problems.**

I. Introdução

Problemas de corte e empacotamento envolvem itens e recipientes regulares e/ou irregulares, de forma que recipientes são cortados para obter itens, ou itens são empacotados em recipientes, para atender a algum objetivo (por exemplo, maximizar a ocupação, minimizar o desperdício, entre outros) [1]. Exemplos de itens/recipientes com formato regular são retângulos, paralepípedos retângulos, círculos, esferas, entre outros. Por outro lado, itens/recipientes de formato irregular são representados por polígonos (convexos e não-convexos) e as partes curvas são comumente aproximadas por um polígono [2]. Uma solução viável para um problema de corte e empacotamento consiste em empacotar os itens sem sobreposição (isto é, não podem ocupar um mesmo espaço) e inteiramente dentro do recipiente (isto é, não é permitida qualquer parte de um item ficar de fora do recipiente).

Problemas de corte e empacotamento que envolvem itens irregulares são comumente chamados de *problemas de nesting* (veja [2]). Os problemas de *nesting* podem ser encontrados em várias situações reais, possuindo aplicação nas indústrias de manufatura, fabricação de vestuário, corte de chapa e fabricação de móveis [3], indústrias de calçados [4], indústrias automobilísticas [5] e de vestuário em couro [6]. Esses problemas, estando geralmente na classe NP-Difícil [7], são encontrados na versão bidimensional em sua grande maioria, existindo poucos trabalhos que envolvem itens/recipientes tridimensionais [8]. Além disso, existe a dificuldade natural ao lidar com a geometria dos itens, pois verificar se existe sobreposição entre itens irregulares é uma tarefa laboriosa do ponto de vista computacional. Segundo [2], as ferramentas até então existentes para lidar com a verificação de sobreposição entre itens eram o método *raster*, a *phi-function*, a trigonometria direta e *no-fit polygon*.

No método *raster*, a representação dos itens e recipientes é feita usando uma matriz e a verificação da sobreposição ocorre sobre essa matriz. Segundo [3], os itens são discretizados em uma malha (matriz), a qual recebe o valor "um" para cada posição. Ao empacotar um item, a malha do recipiente, que possui inicialmente valor "zero" em cada posição, tem as posições que são cobertas pela malha do item atualizadas. Ao verificar que alguma posição da malha do recipiente tem o valor "dois", tem-se o estado de sobreposição entre itens, indicando uma solução inviável. O *no-fit polygon* consiste na determinação de um polígono para cada par de itens, em que um dos itens é fixo e o outro item é móvel. O item móvel translada, sempre encostando, ao redor do item fixo, de forma que um polígono é construído a partir do "caminho" que o item móvel faz. Esse polígono representa o *no-fit polygon* entre os dois itens, tal que o seu interior representa posições que geram o estado de sobreposição [9].

Outras ferramentas surgiram a partir da combinação das ferramentas mencionadas por [2]. Em [10], o *no-fit polygon* foi combinado com o método *raster* para lidar com itens representados por polígonos convexos e não-convexos, resultando no *no-fit raster*. O conceito de *no-fit raster* foi refinado em [11] para tratar itens de qualquer formato, incluindo aqueles formados por curvas complexas. No *no-fit raster*, o *no-fit polygon* entre um par de itens é discretizado em uma matriz preenchida com valor "um" em todas as posições. Em [12], o *no-fit raster* foi utilizado para verificar o estado de sobreposição entre itens, bem como o conceito de *inner-fit*

raster para indicar as posições válidas para empacotar cada item no recipiente.

Com relação aos problemas de *nesting* bidimensionais, em que os itens são representados por polígonos convexos e não-convexos, há poucos resultados para o caso de recipientes fechados (*bins*) conforme [1]. Nessa linha, os problemas principais são do tipo mochila, empacotamento em *bins* fechados e corte de estoque [12]. Os demais problemas envolvem o recipiente com uma, como no caso do empacotamento em faixa, ou duas dimensões abertas [11]. Em problemas de alocação e mochila, busca-se maximizar a ocupação (ou valor empacotado) do recipiente considerando, respectivamente, o empacotamento de itens de um conjunto fracamente e fortemente heterogêneo.

A maioria da literatura de problemas de *nesting* considera o emprego de heurísticas, embora seja possível encontrar a resolução exata de modelos de programação linear inteira em [10], [13], [14], ainda limitados a instâncias de pequeno e médio porte. Por outro lado, tem aparecido um número crescente de heurísticas para esses problemas, incluindo a combinação de heurísticas com a resolução de modelos lineares [15].

A resolução do problema da mochila com itens irregulares foi feita por [16], que propõe um GRASP (*Greedy Randomized Adaptive Search Procedure*), o qual empacota itens irregulares dentro de retângulos objetivando maximizar a taxa de ocupação. Os retângulos, em seguida, são empacotados dentro do recipiente por um algoritmo de programação dinâmica. Em [17] foi proposto um algoritmo construtivo, que empacota itens observando a envoltória convexa resultante da solução parcial, visando minimizar o desperdício no recipiente. A proposta de um modelo de programação linear inteira e de heurísticas construtivas foi feita por [6], considerando recipientes irregulares (e com buracos) cuja representação se deu por uma malha. Recentemente foi desenvolvida em [12] uma heurística geral para resolver diferentes variantes de problemas de *nesting*, incluindo mochila, empacotamento em *bins* e corte de estoque. A heurística geral escolhe aleatoriamente entre seis heurísticas de posicionamento, inspiradas na *bottom-left*, para construir soluções sobre sequências aleatória de itens. Diferentemente de [12], neste trabalho propõe-se um algoritmo genético de chaves aleatórias viciadas para determinar a sequência dos itens, a rotação em que eles devem ser empacotados e a utilização de uma dentre três heurísticas de posicionamento.

Na linha de problemas de empacotamento em faixa, cuja literatura é mais abrangente, pode-se encontrar: uma busca tabu combinada com uma heurística *bottom-left-fill* para o posicionamento dos itens [18]; um algoritmo genético para determinar a sequência em que os itens vão ser empacotados e uma heurística de dois níveis para posicionar os itens sobre uma malha [19]; uma heurística híbrida que combina *cuckoo search* e busca local guiada [20]; um algoritmo genético de chaves aleatórias que usa heurísticas baseadas em *bottom-left* para posicionar os itens [21]; um modelo de programação linear inteira em que os itens são posicionados sobre linhas horizontais [22]; entre outros. Mais recentemente, um algoritmo genético de chaves aleatórias viciadas, que utiliza *bottom-left* para o posicionamento de itens, foi proposto por [11] para o problema com duas dimensões abertas. Diferentemente de [11], o algoritmo genético de chaves aleatórias viciadas proposto neste trabalho considera que os bons indivíduos determinam a probabilidade para que suas informações sejam transmitidas aos seus descendentes.

Devido à existência de relativamente poucos trabalhos na literatura sobre problemas da mochila com itens irregulares, este trabalho desenvolve um algoritmo genético de chaves aleatórias viciadas para o referido problema, devido aos bons resultados obtidos em [11]. Os indivíduos da população contêm informações sobre a ordem e a rotação para empacotar os itens, bem como da heurística de posicionamento que deve ser utilizada. Além disso, se o indivíduo for do grupo elite, o cromossomo também indica a probabilidade de suas informações serem passadas para seus descendentes, o que torna essa probabilidade variável no decorrer das gerações. Três heurísticas de posicionamento são adotadas [12] e consideram que os itens são empacotados em uma malha que representa o recipiente. A partir disto, este artigo está estruturado da seguinte forma: a Seção II descreve o problema e como é verificada a viabilidade da solução; a Seção III descreve as heurísticas de posicionamento, o funcionamento do algoritmo genético e a caracterização dos cromossomos; a Seção IV apresenta e discute os experimentos computacionais realizados sobre um conjunto de instâncias da literatura; e, por fim, a Seção V traz a conclusão e as sugestões para trabalhos futuros.

II. Características do problema

O problema resolvido nesse trabalho consiste na versão bidimensional do Problema da Mochila com itens Irregulares (2PMI), que já é NP-difícil quando os itens são retangulares. São dados: um conjunto de itens I, que podem ser polígonos convexos ou não-convexos, sendo que cada item $i \in I$ é descrito por uma sequência ordenada (no sentido horário) de vértices, podendo incluir buracos; e um recipiente B, que tem o formato retangular, com largura L e comprimento C conhecidos. Cada item $i \in I$ possui um conjunto de rotações R_i, um valor v_i, que neste caso corresponde à sua área, e tem um vértice de referência, que é o vértice localizado na extremidade mais à esquerda, sendo que i é empacotado por este vértice de referência e só pode ser empacotado no máximo uma vez. O objetivo do 2PMI é obter um empacotamento viável de um subconjunto de itens de I de máximo valor. Um empacotamento é viável quando não há sobreposição entre itens e os itens estão inteiramente contidos dentro do recipiente. Note que um empacotamento viável que contém todos os itens de I é sempre ótimo.

Devido à geometria dos itens, a verificação da sobreposição entre itens é feita empregando o *no-fit raster*. Inicialmente, calcula-se o *no-fit polygon* entre todos os pares de itens i e j de I, em cada combinação de suas rotações, utilizando o algoritmo de [23]. Nesse caso, os itens não-convexos são decompostos em polígonos convexos e o algoritmo é aplicado

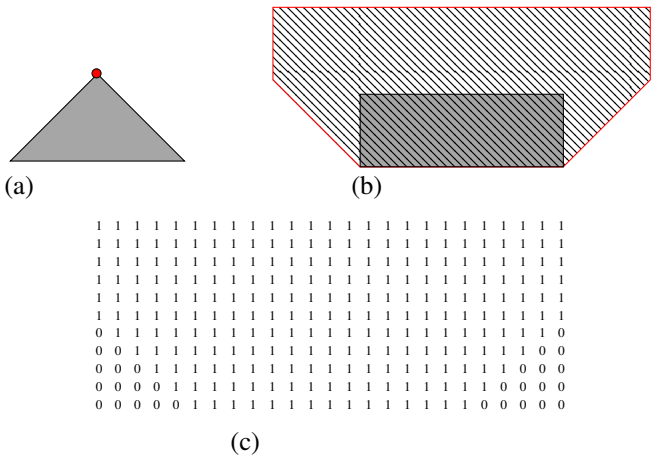

Figure 1. *No-fit raster* entre um retângulo, item fixo, e um triângulo, item orbital.

sobre cada polígono convexo para, então, realizar a união dos *no-fit polygon* parciais e, assim, obter o *no-fit polygon* final entre os itens i e j para a rotação $r \in R_i$ e $s \in R_j$. O *no-fit polygon* entre os itens i e j, para a rotação r e s, é representado por $NFP_{ij,rs}$. Em seguida, constrói-se a matriz correspondente ao $NFP_{ij,rs}$, em que o interior da matriz é preenchido com o valor "um" para indicar que o item j não pode ser empacotado sobre tais posições.

A Figura 1 exemplifica o *no-fit raster* entre os itens i e j. Para discretizar o $NFP_{ij,rs}$ em uma matriz, adota-se uma distância inteira positiva g entre duas coordenadas consecutivas na direção da largura e do comprimento. Quanto menor essa distância, mais próximo do $NFP_{ij,rs}$ se chega, porém, o tamanho da matriz cresce substancialmente, sendo preciso balancear precisão e quantidade de memória computacional necessária para o armazenamento da matriz ao determinar a distância entre as coordenadas.

Para garantir que os itens sempre estejam contidos dentro do recipiente, utiliza-se o *inner-fit raster*. Inicialmente, calcula-se o *inner-fit polygon* para cada item i, em cada uma de suas rotações $r \in R_i$, e o recipiente B, que é o polígono obtido ao transladar i na rotação r, pelo seu vértice de referência, de maneira que i sempre esteja encostando em B, resultando no $IFP_{i,r}$. Em seguida, obtém-se a matriz para o $IFP_{i,r}$ a partir da sua discretização conforme a distância g adotada entre duas coordenadas consecutivas na mesma direção. As posições da matriz em que o item i não pode ser empacotado são preenchidas com o valor "um", uma vez que nessas posições as dimensões do recipiente são extrapoladas. A Figura 2 ilustra o *inner-fit raster* para um item i (triângulo) e um recipiente B (retângulo).

Dado um subconjunto de itens empacotados no recipiente B, para determinar as posições válidas para empacotar outros itens, por exemplo um item j, torna-se suficiente calcular a interseção entre os *no-fit rasters* dos itens já empacotados e do item escolhido para ser empacotado. As posições dessa matriz resultante que estão preenchidas com o valor "zero"

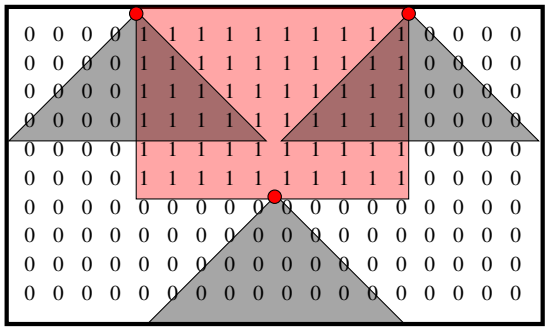

Figure 2. *Inner-fit raster* de um triângulo em um recipiente retangular.

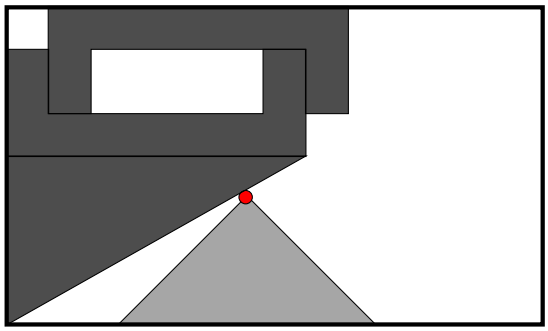

Figure 3. Empacotamento de um triângulo em um recipiente retangular parcialmente preenchido.

indicam as posições em que j pode ser empacotado em B sem gerar sobreposição com qualquer um dos itens já empacotados. A Figura 3 ilustra o caso em que existe um item i (retângulo) já empacotado em B e deseja-se empacotar um item j (triângulo).

III. Heurística proposta

O algoritmo genético de chaves aleatórias viciadas (do inglês *Biased Random-Key Genetic Algorithm - BRKGA* [24]) é uma meta-heurística evolutiva que traz para o algoritmo genético o conceito de chaves aleatórias, para a representação de indivíduos [25], além de considerar a ideia de grupos elites e não-elites e, assim, privilegiar a obtenção de informações de indivíduos elites. Um algoritmo genético possui uma população de indivíduos, também identificados como cromossomos, os quais passam por um processo sucessivo de evolução, gerando novas populações a partir de etapas como seleção, cruzamento e mutação.

No BRKGA, cada cromossomo é representado por um vetor c de tamanho n, cujas componentes são chaves aleatórias. A população inicial consiste de P cromossomos gerados aleatoriamente. Em seguida, cada cromossomo $c \in P$ passa por um decodificador, em que o vetor de chaves aleatórias é transformado em uma solução do problema e, assim, obtém-se a sua aptidão calculando $f(c)$. Com isso, a população é dividida em dois grupos, um elite P_e e outro não-elite $P_{\bar{e}}$, tal que $P = P_e \cup P_{\bar{e}}$. O grupo elite contém os melhores indivíduos, ou seja, aqueles com melhor valor de aptidão $f(c)$.

Para criar a população P da próxima geração do BRKGA, mantém-se o grupo P_e sem modificações em P. O restante dos indivíduos em P advém de dois outros grupos, P_m e P_f. O grupo P_m representa os indivíduos mutantes, que são cromossomos gerados aleatoriamente, visando introduzir diversidade na população e evitar convergência prematura para ótimos locais. O grupo P_f consiste dos indivíduos oriundos por meio do cruzamento.

A etapa de cruzamento no BRKGA consiste em selecionar, aleatoriamente e com reposição, indivíduos do conjunto P_e e $P_{\bar{e}}$ para gerar os indivíduos descendentes em P_f. Um indivíduo descendente $c_d \in P_f$ é gerado a partir de um indivíduo $c_e \in P_e$ e outro $c_{\bar{e}} \in P_{\bar{e}}$, utilizando a combinação uniforme parametrizada e uma probabilidade viciada $\rho_e > 1/2$ de herdar as chaves de c_e. Assim, a componente $c_d[i]$ advém de $c_e[i]$ com probabilidade ρ_e ou de $c_{\bar{e}}$ com probabilidade $1-\rho_e$. A população P da próxima geração é, então, composta da união entre P_e, P_m e P_f, sendo sempre de tamanho constante durante o processo de evolução.

Os valores utilizados pelo BRKGA para os seus parâmetros são constantes e fornecidos como entrada. São eles: quantidade de chaves n, quantidade máxima de gerações max, tamanho da população P, tamanho da população elite P_e, tamanho da população mutante P_m, probabilidade de herdar chaves de pais elites ρ_e. Observa-se que o BRKGA necessita de um bom decodificador e da correta calibração de seus parâmetros para que boas soluções sejam obtidas ao final do processo evolutivo.

A partir da estrutura de BRKGA descrita anteriormente e da interface desenvolvida em [26], propõe-se um decodificador para o 2PMI e considera-se que a probabilidade ρ_e não é um valor constante, mas sim uma característica do indivíduo, isto é, é uma chave aleatória no cromossomo.

Para o 2PMI, cada cromossomo c possui n chaves, em que $n = |I| + 2$, isto é, n contempla o total de itens em I, enquanto as outras duas chaves são para indicar qual heurística de posicionamento, dentre três heurísticas disponíveis, utilizar para c e qual a probabilidade ρ_e deve ser utilizada caso c seja um indivíduo elite selecionado para gerar um descendente. As componentes de 1 até $n-2$ de c estão indexadas com os itens I, tal que a chave aleatória da posição $i \in 1, 2, \ldots, n-2$ representa o respectivo item $i = 1, 2, \ldots, |I|$. O decodificador para o 2PMI está descrito no Algoritmo 1 cujos parâmetros de entrada são um cromossomo, o conjunto de itens, o recipiente e a distância utilizada para a discretização. Com isso, denomina-se o BRKGA desenvolvido para o 2PMI como BRKGA$_{2PMI}$.

O Algoritmo 1 inicia copiando as $n-2$ primeiras chaves aleatórias do cromossomo c para um vetor Or. Essas chaves estão associadas sequencialmente com os itens de I. Em seguida, o vetor Or é ordenado de forma não decrescente observando as chaves aleatórias. Observe que ao realizar essa ordenação, a sequência inicial dos itens é modificada para uma nova sequência. Na Linha 5, faz-se a chamada de uma heurística de posicionamento para determinar a lista ordenada de pontos LP onde os itens podem ser empacotados, sendo que a escolha da heurística depende da chave aleatória em $c[n-1]$.

Algoritmo 1: Decodificador de cromossomos do BRKGA$_{2PMI}$

Entrada: cromossomo c de tamanho n, conjunto de itens I, recipiente $B = (L, C)$, distância g.
Saída: Sol, que é um empacotamento viável, e Ar, que é o valor deste empacotamento viável.

```
1  início
2      Ar ← 0;
3      Or[...] ← c[1, ..., n − 2];
4      Or ← ordenar Or de forma não decrescente pelo
         valor de chave aleatória;
5      LP ← aplicar a heurística de posicionamento dado o
         intervalo que c[n − 1] está:
         [0, 1/3), [1/3, 2/3), [2/3, 1);
6      j ← 1;
7      enquanto j ≤ n − 2 faça
8          i ← o item de I associado à chave aleatória
             Or[j];
9          r ← a rotação de i dado o intervalo em que
             Or[j] está: [0, 1/|R_i|), [1/|R_i|, 2/|R_i|), ...;
10         para cada ponto (a, b) ∈ LP conforme a ordem
             dada pela heurística de posicionamento faça
11             se empacotamento é viável para i, na rotação
                 r, em (a, b) então
12                 Sol ← Sol ∪ {i, r, (a, b)};
13                 Ar ← Ar + v_i;
14                 Remover de LP os pontos sobrepostos ao
                     empacotar i, na rotação r, em (a, b);
15                 j ← j + 1;
16                 Ir para a Linha 7;
17         Atualizar r para uma rotação em R_i que ainda
             não foi testada e Voltar para a Linha 10;
18     retorna {Sol, Ar}
```

Os itens no Algoritmo 1 são empacotados seguindo a sequência em Or e a ordem dos pontos em LP, conforme o laço das Linhas 7-17. Nas Linhas 8 e 9, respectivamente, o item i é identificado e a primeira rotação r em que ele deve ser empacotado é obtida da chave aleatória associada a i, que está na posição $Or[j]$. O número de rotações para i, que é $|R_i|$, indica em quantos subintervalos consecutivos o intervalo $[0, 1)$ vai ser particionado. Por exemplo, caso o item i possa ser empacotado nas rotações 0 e 90 graus, então $|R_i| = 2$ e, assim, há dois intervalos: $[0, 1/2)$, que está associado à rotação de 0 grau, e $[1/2, 2/2)$, que está associado à rotação de 90 graus.

Na Linha 11 do Algoritmo 1, testa-se a viabilidade de empacotar o item i, dada a rotação r, em um ponto (a, b) da lista ordenada LP. Os pontos da lista são percorridos na ordem em que eles foram inseridos em LP, tal que se busca pelo primeiro ponto viável para o empacotamento. Caso não haja qualquer ponto viável para i na rotação r, tenta-se novamente para uma outra rotação r em R_i (Linha 17), até considerar todas as rotações em R_i. Caso não seja possível empacotar i

em qualquer uma de suas rotações, o item i é deixado de fora. Quando é possível empacotar i em alguma rotação r, atualiza-se a solução Sol (Linha 12) e o valor do empacotamento em Ar (Linha 13). Nas Linhas 14, 15 e 16, os pontos da lista LP que se tornam inviáveis após o empacotamento de i são removidos e o algoritmo volta para o início do laço na Linha 7.

Com relação à chave aleatória na posição $c[n-1]$ na Linha 5 Algoritmo 1, caso ela esteja em $[0, 1/3)$, seleciona-se a primeira heurística de posicionamento (que é a *bottom-left* descrita no Algoritmo 2), caso ela esteja em $[1/3, 2/3)$, seleciona-se a segunda heurística de posicionamento (que é a *horizontal zig-zag* descrita no Algoritmo 3), caso contrário, seleciona-se a terceira heurística de posicionamento (que é a *left-bottom* descrita no Algoritmo 4). As heurísticas de posicionamento devolvem uma lista ordenada de pontos LP com a sequência de pontos que deve ser seguida para empacotar os itens dentro do recipiente.

No Algoritmo 2, a lista de pontos está ordenada de forma a indicar que os itens sejam empacotados mais à esquerda e abaixo possível dentro do recipiente. No Algoritmo 3, a lista de pontos segue um zig-zag horizontal para alocar os itens, indo da esquerda para a direita se o comprimento b é par e, caso contrário, indo da direita para a esquerda. No Algoritmo 4, a ordem dos pontos na lista indica que os itens sejam empacotados mais abaixo e à esquerda possível. Outras heurísticas de posicionamento foram propostas e comparadas em [12], porém algumas com desempenho não muito satisfatório.

Algoritmo 2: Bottom-Left

Entrada: Recipiente $B = (L, C)$, distância g.
Saída: Lista ordenada de pontos LP associada ao recipiente.

```
1  início
2      LP ← ∅;
3      para a ∈ {0, g, 2g, 3g, ..., L − 1} faça
4          para b ∈ {0, g, 2g, 3g, ..., C − 1} faça
5              LP ← LP ∪ {a, b};
6      retorna LP
```

A chave aleatória na última posição do cromossomo c é utilizada somente na etapa de cruzamento e no caso em que c faz parte do grupo elite e foi selecionado para gerar um indivíduo descendente. Assim, ao invés de adotar a probabilidade ρ_e como uma constante, utiliza-se o valor da chave aleatória em $c[n]$ para ser a probabilidade do descendente herdar chaves aleatórias do cromossomo c. Experimentos computacionais preliminares mostraram que boas soluções podem ser obtidas rapidamente ao tornar a probabilidade ρ_e dinâmica e dependente do cromossomo.

IV. Resultados experimentais

O BRKGA$_{2PMI}$ foi codificado em linguagem C++ e os experimentos foram realizados em um computador com processador Intel Core i5 2.9 GHz, 8 GB de RAM e sistema macOS High Sierra. Os parâmetros adotados para o BRKGA$_{2PMI}$ foram obtidos a partir de testes preliminares, observando os intervalos para: o número de máximo de gerações, $max \in [5000, 15000]$; tamanho da população, $P \in [1000, 2000]$; tamanho da população elite $P_e \in [0,2P,\ 0,35P]$; e, tamanho da população mutante, $P_m \in [0,1P,\ 0,25P]$. Estabeleceu-se um tempo de limite máximo de 600 segundos para a execução do BRKGA$_{2PMI}$ para resolver cada instância, sendo que o BRKGA$_{2PMI}$ pode finalizar caso não tenha atingido max, porém tenha atingido o tempo limite. Além disso, o BRKGA$_{2PMI}$ também pode finalizar quando consegue empacotar todos itens da instância. A partir do teste de calibração (via tentativa e erro), obteve-se os seguintes parâmetros: $max = 10000$, $P = 1500$, $P_e = 0,3P$ e $P_m = 0,2P$.

As instâncias utilizadas nos experimentos foram as mesmas adotadas por [16], uma vez que elas foram disponibilizadas pelos autores na Internet[1], além de permitir uma comparação direta entre o BRKGA$_{2PMI}$ e o GRASP proposto por tais autores. Não foi realizada uma comparação com modelos matemáticos, por não se conhecer na literatura resultados envolvendo modelos para o 2PMI. Com isso, há um total de 15 instâncias cujos dados estão na Tabela I: nome, quantidade total de itens, dimensões do recipiente B, rotações (em graus)

Algoritmo 3: Horizontal Zig-Zag

Entrada: Recipiente $B = (L, C)$, distância g.
Saída: Lista ordenada de pontos LP associada ao recipiente.

```
1  início
2      LP ← ∅;
3      para b ∈ {0, g, 2g, 3g, ..., C − 1} faça
4          se b mod 2 = 0 então
5              para a ∈ {0, g, 2g, 3g, ..., L − 1} faça
6                  LP ← LP ∪ {a, b};
7          senão
8              para a ∈ {0, g, 2g, 3g, ..., L − 1} faça
9                  LP ← LP ∪ {a, b};
10     retorna LP
```

Algoritmo 4: Left-Bottom

Entrada: Recipiente $B = (L, C)$, distância g.
Saída: Lista ordenada de pontos LP associada ao recipiente.

```
1  início
2      LP ← ∅;
3      para b ∈ {0, g, 2g, 3g, ..., C − 1} faça
4          para a ∈ {0, g, 2g, 3g, ..., L − 1} faça
5              LP ← LP ∪ {a, b};
6      retorna LP
```

[1] http://www.loco.ic.unicamp.br/instances/2dik.html

Table I
DADOS DAS INSTÂNCIAS.

Nome	Total de itens	Dimensões (L, C)	Rotações	Escala
Albano	24	(4900, 10122,63)	(0, 180)	1:10
Dagli	30	(60, 65,6)	(0, 180)	1:0,1
Dighe1	16	(100, 138,13)	(0)	1:1
Dighe2	10	(100, 134,5)	(0)	1:1
Fu	12	(38, 34)	(0, 90, 180)	1:1
Jakobs1	25	(40, 13)	(0, 90, 180)	1:1
Jakobs2	25	(70, 28,2)	(0, 90, 180)	1:1
Mao	20	(2550, 2058,6)	(0, 90, 180)	1:10
Marques	24	(104, 83,6)	(0, 90, 180)	1:1
Shapes0	43	(40, 63)	(0)	1:1
Shapes1	43	(40, 59)	(0, 180)	1:1
Shapes2	28	(15, 27,3)	(0, 180)	1:1
Shirts	99	(40, 63,13)	(0, 180)	1:1
Swim	48	(5752, 6568)	(0, 180)	1:5
Trousers	64	(79, 245,75)	(0, 180)	1:1

Table II
RESULTADOS DO BRKGA$_{2PMI}$ E GRASP SOBRE AS 15 INSTÂNCIAS.

	BRKGA		GRASP [16]		
Nome	Tempo (s)	Taxa de Ocupação (%)	Tempo (s)	Taxa de Ocupação (%)	Desvio (%)
Albano	316,01	0,8038	961,59	0,8038	0,0
Dagli	268,26	**0,7731***	1106,40	0,7586	1,9
Dighe1	6,46	0,7240*	10,68	0,7240*	0,0
Dighe2	3,30	0,7460*	0,07	0,7460*	0,0
Fu	1,36	0,8382*	21,79	0,8382*	0,0
Jakobs1	1,36	0,7538*	8,30	0,7538*	0,0
Jakobs2	4,57	0,6844*	565,71	0,6844*	0,0
Mao	6,79	0,7160*	120,42	0,7160*	0,0
Marques	16,14	0,8274*	217,77	0,8274*	0,0
Shapes0	0,89	**0,6222**	1552,46	0,6016	3,4
Shapes1	1,47	**0,6593**	3891,60	0,6424	2,6
Shapes2	0,27	**0,7558**	1048,12	0,7289	3,7
Shirts	3,06	**0,8419**	14317,13	0,7702	10,0
Swim	3,79	**0,6623**	39781,43	0,6427	4,8
Trousers	21,73	**0,8384**	5796,59	0,7866	6,5

permitidas para os itens e a escala adotada para a discretização e construção das malhas de *no-fit rasters* e *no-fit polygons*. Com relação às dimensões do recipiente, conforme a escala adotada, considerou-se apenas a parte inteira dos valores, isto é, arrendondaram-se os valores para baixo.

Os resultados reportados na Tabela II foram obtidos a partir da execução do BRKGA$_{2PMI}$ 5 vezes sobre cada instância. Cada linha da tabela contém: nome da instância; tempo (em segundos) e taxa de ocupação para a melhor solução que o BRKGA$_{2PMI}$ encontrou entre as 5 execuções; tempo (em segundos) e taxa de ocupação obtidos pelo GRASP de [16] considerando 15 execuções; e, desvio relativo (em porcentagem) da taxa ocupação do BRKGA$_{2PMI}$ com o GRASP. As soluções marcadas com um '*' significam que todos os itens foram empacotados no recipiente, ou seja, trata-se de uma solução ótima. Com relação à discretização, percebeu-se que para 3 instâncias (Albano, Mao e Swim), uma escala maior (isto é, 1 ponto a cada 10, ou 5, unidades de distância) foi suficiente para o BRKGA$_{2PMI}$ trazer soluções melhores do que o GRASP. Por outro lado, para a instância Dagli foi necessário considerar uma escala menor, de 0,1:1, isto é, com 1 ponto a cada 0,1 unidades. Nas demais instância, a escala adotada foi de 1 ponto a cada unidade de distância.

Ao observar os resultados na Tabela II, nota-se que tanto o BRKGA$_{2PMI}$ quanto o GRASP conseguiram empacotar todos os itens das instâncias, isto é, obter uma solução sabidamente ótima, para 7 das 15 instâncias (Dighe1, Dighe2, Fu, Jakobs1, Jakobs2, Mao, Marques). Nas demais instâncias, o BRKGA$_{2PMI}$ empata com o GRASP em uma instância (Albano) e é superior ao GRASP, conseguindo empacotar todos os itens (isto é, solução sabidamente ótima), para outra instância (Dagli), aumentando a sua taxa de ocupação em 1,9%. O BRKGA$_{2PMI}$ ainda consegue melhorar a taxa de ocupação em 4,8%, na média, para as instâncias Shapes0, Shapes1, Shapes2, Shirts, Swim, Trousers. O melhor resultado em termos de ocupação foi para a instância Shirts, sendo que o BRKGA$_{2PMI}$ aumentou a taxa de ocupação em 9,3% ao comparar com o GRASP.

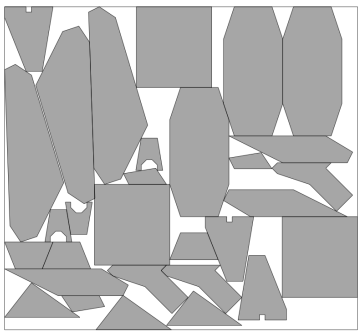

Figure 4. Solução para a instância Dagli.

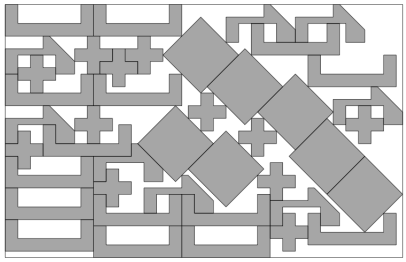

Figure 5. Solução para a instância Shapes0.

Embora os computadores utilizados nos experimentos para o BRKGA$_{2PMI}$ e o GRASP sejam diferentes, o que dificulta uma comparação justa entre os tempos reportados pelos algoritmos, observa-se que o tempo do BRKGA$_{2PMI}$ foi inferior ao do GRASP em 14 das 15 instâncias, sendo o desvio percentual de redução de 88,9%, na média. Apenas na instância Dighe2 que o tempo do BRKGA$_{2PMI}$ foi levemente superior ao do GRASP, porém em instâncias como Shirts e Swim o BRKGA$_{2PMI}$ precisou de poucos segundos ao passo que o GRASP precisou de mais de 3 horas de processamento. Nota-se ainda que o BRKGA$_{2PMI}$ conseguiu obter soluções iguais ou superiores às do GRASP para 13 instâncias em tempo inferior a 30 segundos. As Figuras 4 a 10 apresentam as soluções das instâncias que o BRKGA$_{2PMI}$ conseguiu melhorar a taxa de ocupação.

V. CONCLUSÕES E TRABALHOS FUTUROS

O problema da mochila bidimensional com itens irregulares foi resolvido por um algoritmo genético de chaves aleatórias viciadas, denominado BRKGA$_{2PMI}$, uma vez que o BRKGA

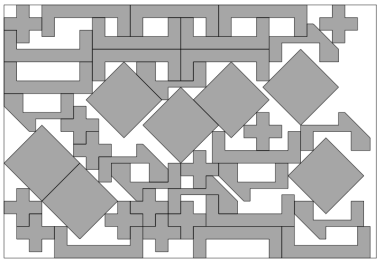

Figure 6. Solução para a instância Shapes1.

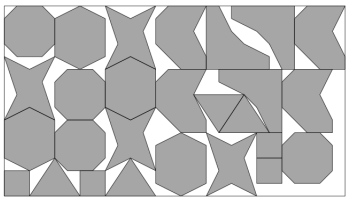

Figure 7. Solução para a instância Shapes2.

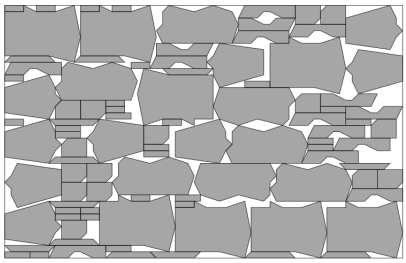

Figure 8. Solução para a instância Shirts.

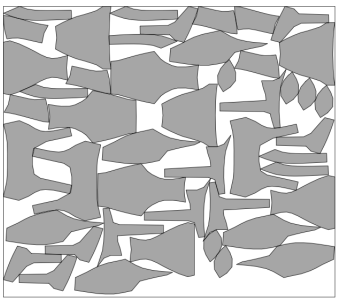

Figure 9. Solução para a instância Swim.

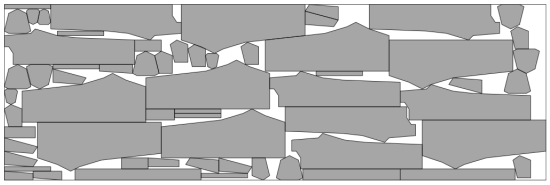

Figure 10. Solução para a instância Trousers.

já foi empregado para resolver outros problemas de *nesting* e trouxe resultados bem competitivos. No BRKGA$_{2PMI}$, o cromossomo carrega informações sobre a sequência e a rotação em que os itens devem ser alocados no recipiente, qual heurística de posicionamento deve ser utilizada e qual a taxa, caso o cromossomo seja elite, que suas informações serão passadas para os seus descendentes.

Os resultados que o BRKGA$_{2PMI}$ obteve para 15 instâncias da literatura foram superiores ao de uma estratégia GRASP em termos de taxa de ocupação do recipiente e em tempo computacional (em particular, nas instâncias mais difíceis). O BRKGA$_{2PMI}$ melhorou a taxa de ocupação de 7 instâncias, com um percentual médio de aumento de 4,2%, teve um tempo computacional inferior em 14 instâncias (embora os computadores utilizados sejam diferentes) e conseguiu a solução sabidamente ótima para 8 instâncias. Além disso, o GRASP não conseguiu superar os resultados, em termos de ocupação, para nenhuma das 15 instâncias.

Os trabalhos futuros devem focar na proposta de uma busca local, uma vez que o BRKGA$_{2PMI}$ estagnou em ótimos locais nas primeiras gerações para algumas instâncias. Pretende-se também estender o BRKGA$_{2PMI}$ para resolver outras variantes de problemas de corte e empacotamento de itens irregulares na versão bidimensional, como empacotamento em *bins* e corte de estoque.

AGRADECIMENTOS

Os autores agradecem o apoio financeiro fornecido pela CAPES, CNPq (409043/2016-8) e FAPESP (2013/07375-0 e 2016/01860-1).

REFERENCES

[1] G. Wäscher, H. Haußner, and H. Schumann, "An improved typology of cutting and packing problems," *European Journal of Operational Research*, vol. 183, no. 3, pp. 1109 – 1130, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S037722170600292X

[2] J. A. Bennell and J. F. Oliveira, "The geometry of nesting problems: A tutorial," *European Journal of Operational Research*, vol. 184, no. 2, pp. 397 – 415, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221706012379

[3] ——, "A tutorial in irregular shape packing problems," *Journal of the Operational Research Society*, vol. 60, pp. 93–105, 2009.

[4] H.-H. Yang and C.-L. Lin, "On genetic algorithms for shoe making nesting – a taiwan case," *Expert Systems with Applications*, vol. 36, no. 2, Part 1, pp. 1134 – 1141, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417407005192

[5] C. Alves, P. Brás, J. V. de Carvalho, and T. Pinto, "New constructive algorithms for leather nesting in the automotive industry," *Computers & Operations Research*, vol. 39, no. 7, pp. 1487 – 1505, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S030505481100253X

[6] R. Baldacci, M. A. Boschetti, M. Ganovelli, and V. Maniezzo, "Algorithms for nesting with defects," *Discrete Applied Mathematics*, vol. 163, Part 1, pp. 17 – 33, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0166218X12001308

[7] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979.

[8] J. Egeblad, B. K. Nielsen, and A. Odgaard, "Fast neighborhood search for two- and three-dimensional nesting problems," *European Journal of Operational Research*, vol. 183, no. 3, pp. 1249 – 1266, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S037722170600302X

[9] E. Burke, R. Hellier, G. Kendall, and G. Whitwell, "Complete and robust no-fit polygon generation for the irregular stock cutting problem," *European Journal of Operational Research*, vol. 179, no. 1, pp. 27 – 49, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221706001639

[10] F. M. Toledo, M. A. Carravilla, C. Ribeiro, J. F. Oliveira, and A. M. Gomes, "The dotted-board model: A new mip model for nesting irregular shapes," *International Journal of Production Economics*, vol. 145, no. 2, pp. 478 – 487, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925527313001722

[11] L. R. Mundim, M. Andretta, and T. A. de Queiroz, "A biased random key genetic algorithm for open dimension nesting problems using no-fit raster," *Expert Systems with Applications*, vol. 81, pp. 358 – 371, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417417302233

[12] L. R. Mundim, M. Andretta, M. A. Carravilla, and J. F. Oliveira, "A general heuristic for two-dimensional nesting problems with limited-size containers," *International Journal of Production Research*, vol. 56, no. 1-2, pp. 709–732, 2018. [Online]. Available: https://doi.org/10.1080/00207543.2017.1394598

[13] R. Alvarez-Valdes, A. Martinez, and J. Tamarit, "A branch & bound algorithm for cutting and packing irregularly shaped pieces," *International Journal of Production Economics*, vol. 145, no. 2, pp. 463 – 477, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925527313001692

[14] L. H. Cherri, L. R. Mundim, M. Andretta, F. M. Toledo, J. F. Oliveira, and M. A. Carravilla, "Robust mixed-integer linear programming models for the irregular strip packing problem," *European Journal of Operational Research*, vol. 253, no. 3, pp. 570 – 583, 2016. [Online]. Available: //www.sciencedirect.com/science/article/pii/S0377221716301370

[15] A. M. Gomes and J. F. Oliveira, "Solving irregular strip packing problems by hybridising simulated annealing and linear programming," *European Journal of Operational Research*, vol. 171, no. 3, pp. 811 – 829, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221704005879

[16] A. M. D. Valle, T. A. Queiroz, F. K. Miyazawa, and E. C. Xavier, "Heuristics for two-dimensional knapsack and cutting stock problems with items of irregular shape," *Expert Systems with Applications*, vol. 39, no. 16, pp. 12 589 – 12 598, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417412007233

[17] D. Dalalah, S. Khrais, and K. Bataineh, "Waste minimization in irregular stock cutting," *Journal of Manufacturing Systems*, vol. 33, no. 1, pp. 27 – 40, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0278612513001209

[18] E. Burke, R. Hellier, G. Kendall, and G. Whitwell, "A new bottom-left-fill heuristic algorithm for the two-dimensional irregular packing problem," *Operations Research*, vol. 54, no. 3, pp. 587–601, 2006. [Online]. Available: http://dx.doi.org/10.1287/opre.1060.0293

[19] W. Wong, X. Wang, P. Mok, S. Leung, and C. Kwong, "Solving the two-dimensional irregular objects allocation problems by using a two-stage packing approach," *Expert Systems with Applications*, vol. 36, no. 2, Part 2, pp. 3489 – 3496, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417408001073

[20] A. Elkeran, "A new approach for sheet nesting problem using guided cuckoo search and pairwise clustering," *European Journal of Operational Research*, vol. 231, no. 3, pp. 757 – 769, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221713005080

[21] P. R. Pinheiro, B. A. Júnior, and R. D. Saraiva, "A random-key genetic algorithm for solving the nesting problem," *International Journal of Computer Integrated Manufacturing*, vol. 0, no. 0, pp. 1–7, 2015. [Online]. Available: http://dx.doi.org/10.1080/0951192X.2015.1036522

[22] A. A. L. ao, F. M. Toledo, J. F. Oliveira, and M. A. Carravilla, "A semi-continuous mip model for the irregular strip packing problem," *International Journal of Production Research*, vol. 54, no. 3, pp. 712–721, 2016. [Online]. Available: https://doi.org/10.1080/00207543.2015.1041571

[23] R. Cuninghame-Green, "Geometry, shoemaking and the milk tray problem," *New Scientist*, vol. 1677, pp. 50–53, 1989. [Online]. Available: https://www.newscientist.com/article/mg12316773-700/

[24] J. F. Gonçalves and M. G. C. Resende, "Biased random-key genetic algorithms for combinatorial optimization," *Journal of Heuristics*, vol. 17, no. 5, pp. 487–525, Oct 2011. [Online]. Available: https://doi.org/10.1007/s10732-010-9143-1

[25] J. C. Bean, "Genetic algorithms and random keys for sequencing and optimization." *INFORMS Journal on Computing*, vol. 6, no. 2, pp. 154–160, 1994. [Online]. Available: http://dblp.uni-trier.de/db/journals/informs/informs6.html\#Bean94

[26] R. Toso and M. Resende, "A c++application programming interface for biased random-key genetic algorithms," *Optimization Methods and Software*, vol. 30, no. 1, pp. 81–93, 2015. [Online]. Available: http://dx.doi.org/10.1080/10556788.2014.890197