CAPÍTULO 5

O BRKGA APLICADO EM PROBLEMAS DE CORTE DE ITENS IRREGULARES EM UM ÚNICO RECIPIENTE

Leandro Resende Mundim
Universidade de São Paulo,
Instituto de Ciências Matemáticas
e de Computação.
mundim@icmc.usp.br

Thiago Alves de Queiroz
Universidade Federal de Goiás,
Instituto de Matemática e Tecnologia.
taq@ufg.br

Marina Andretta
Universidade de São Paulo,
Instituto de Ciências Matemáticas
e de Computação.
andretta@icmc.usp.br

RESUMO

Os problemas de corte de itens irregulares (polígonos convexos e não convexos) em um único recipiente são encontrados em diversas indústrias, tais como a têxtil, a de calçados, a de móveis e outras. Na versão bidimensional existem dois problemas básicos: o corte de itens em faixa e o corte de itens em placas.

Neste trabalho, visando a resolver ambos os problemas, foi proposta uma abordagem baseada no algoritmo genético de chaves aleatórias viciadas, partindo de um *framework* proposto na literatura. O diferencial está na técnica de alocação, que combina os cantos do recipiente, utilizando o canto inferior esquerdo ou o canto superior esquerdo. Para encontrar posições viáveis e evitar a sobreposição, foi usada uma malha de pontos extraída da técnica de *no-fit polygon*. Os experimentos computacionais demostraram que o algoritmo proposto é competitivo, obtendo resultados ótimos para ambos os problemas, além de melhorar resultados recentes da literatura.

Palavras-chaves: Problema de corte em faixa, Problema de corte em placas, Itens irregulares, Algoritmo genético, Chaves viciadas, *No-fit polygon*.

1 INTRODUÇÃO

Problemas de corte e de empacotamento de itens irregulares (polígonos convexos e não convexos), também conhecidos como problemas de *nesting*, são problemas em que um conjunto de itens irregulares deve ser cortado, sem sobreposição, a partir de um recipiente maior, em geral visando a minimizar o desperdício do recipiente utilizado, pela classificação de Wäscher et al. (2007). Esses problemas aparecem na área de otimização combinatória e com muita frequência na indústria têxtil, durante o corte de peças de roupa e tecido. Outras aplicações, como cortar placas de metal, peças de couro e cortes de espuma, aparecem nas indústrias metal-mecânica, calçadista e moveleira, por exemplo. As transportadoras enfrentam esse problema ao terem que empacotar produtos sem sobreposição em contêineres e durante o transporte de estátuas e peças de museu. Algumas aplicações desses problemas são apresentadas em: Yang e Lin (2009) (indústria do calçado), Alves et al. (2012) (indústria automotiva), Lee et al. (2013) (indústria de construção naval) e, mais recentemente, Baldacci et al. (2014) (indústrias de vestuário de couro e mobiliário).

Além da importância econômica (redução de custos), esse problema tem grande importância ambiental. Ao reduzir o desperdício do material utilizado, melhora-se a utilização de recursos naturais, tais como o alumínio, o couro, a madeira e o vidro. Com a intenção de melhorar a utilização desses recursos, diversas técnicas de resolução para problemas de corte de itens irregulares vêm sendo desenvolvidas, em geral heurísticas e meta-heurísticas. A grande maioria de trabalhos heurísticos reflete a real dificuldade do problema, escondida por sua aparente simplicidade. Para mais detalhes sobre a prova de que os problemas com itens irregulares (ou problemas de *nesting*) são NP-difíceis, ver Nielsen e Odgaard (2003).

Este trabalho considera dois problemas de corte de itens irregulares em sua versão bidimensional: o problema de corte de itens irregulares em faixa e o pro-

blema de corte de itens irregulares em placas. O corte de itens em faixa é um problema de dimensão aberta, sendo referenciado no inglês por *irregular strip packing problem*. Nesse problema, todos os itens devem ser alocados em um recipiente com largura conhecida e comprimento tido como "infinito", com o objetivo de alocar todos os itens utilizando o menor comprimento de faixa possível. Já o problema de corte de placas é um problema de dimensões fixas, no qual a partir de uma placa (ou recipiente) com dimensões conhecidas se deseja maximizar a utilização do recipiente pelo corte de um subconjunto de itens. Os trabalhos que lidam com o problema de corte de placas são conhecidos na literatura como: problema de alocação (do inglês, *placement problem*) e problema da mochila (do inglês, *knapsack problem*). Por simplicidade, neste trabalho denota-se por problema bidimensional de corte de itens irregulares em faixa (2PCF) e problema bidimensional de corte de itens irregulares em placas (2PCP).

Para o 2PCF, alguns trabalhos e estratégias se destacam, especialmente pela sua simplicidade e pelos bons resultados apresentados. Essas estratégias construtivas se baseiam na inserção de itens iterativamente, até alocar todos os itens no recipiente ou não ser mais possível alocar qualquer item. Nesse sentido, Oliveira et al. (2000) apresentaram um algoritmo construtivo definido sob cinco critérios de ordenação dos itens e vários critérios de alocação, de forma a transformar a sequência dos itens em um leiaute viável. Esse trabalho serviu de base para vários outros que se basearam nas mesmas estratégias de alocação e critérios de ordenação. Outras heurísticas construtivas bastante utilizadas são aquelas em que é dada uma sequência para alocar os itens, utilizando a estratégia de alocar cada item da sequência o mais próximo do canto inferior esquerdo (conhecida como *bottom-left*), como em Dowsland et al. (2002), Gomes e Oliveira (2002) e Burke et al. (2006).

Jakobs (1996) apresentou um dos primeiros algoritmos genéticos para o 2PCF. O autor criou duas instâncias artificiais e aplicou o algoritmo genético para alocar a envoltória retangular dos itens por meio de uma estratégia *bottom-left*. Em seguida, o método aplica sobre a envoltória real dos itens uma compactação em linha para melhorar a solução. A ordem com que os itens são alocados pela heurística *bottom-left* é dada pela sequência em que os itens estão dispostos no cromossomo. Ainda no mesmo problema, a estratégia híbrida proposta por Gomes e Oliveira (2006) melhorou vários resultados da literatura. Ela consiste em uma meta-heurística de recozimento simulado, que utiliza modelos lineares de separação e compactação. Essa estratégia utiliza a técnica de *bottom-left* discreta para gerar um leiaute inicial. Recentemente, Elkeran (2013) combinou a meta-heurística *cuckoo search* de Yang e Deb (2010) com uma busca local direcionada na tentativa de escapar de mínimos locais. Também foram propostos modelos matemáticos para a compactação do leiaute e algoritmos de separação para retirar as possíveis sobreposições, sendo possível melhorar vários resultados existentes. Outras meta-heurísticas que obtiveram

sucesso para resolver o 2PCF foram o *iterated local search* de Imamichi et al. (2009) e a busca local guiada de Umetani et al. (2009).

Existem alguns trabalhos que utilizam métodos exatos para resolver o 2PCF. Dentre eles, destacam-se o método baseado em programação por restrições de Carravilla et al. (2003) e o modelo de programação inteira mista de Fischetti e Luzzi (2009). Alvarez-Valdes et al. (2013) reformularam o modelo de Fischetti e Luzzi (2009) e propuseram um algoritmo *branch-and-bound* para o 2PCF, além de estender o modelo de compactação de Gomes e Oliveira (2006). Toledo et al. (2013) apresentaram um modelo inteiro misto e resolveram as maiores instâncias da literatura para algoritmo exatos, entretanto a solução é ótima para uma malha de pontos. Recentemente, Leão et al. (2015) propuseram um modelo inteiro misto para o corte de itens irregulares no domínio semicontínuo, ou seja, os itens podem ser alocados em qualquer ponto sobre um conjunto fixo de linhas. Esse trabalho é o primeiro combinar os modelos contínuos e discretos.

Para o 2PCP, o número de trabalhos ainda é reduzido. Trabalhos recentes que envolvem variações do problema descrito podem ser encontrados em Del Valle (2010), Del Valle et al. (2012), Mundim e Queiroz (2012), Silveira (2013) e Dalalah et al. (2014). Del Valle (2010) propôs um algoritmo híbrido para o problema de alocação irrestrito, no qual os itens têm demanda ilimitada e uma heurística baseada em GRASP para o 2PCP. Uma solução inicial é obtida de forma gulosa, alocando, dentre uma lista aleatória de itens não alocados, o item que tenha o melhor custo/benefício. Esse custo depende da ocupação retangular e da área da envoltória retangular do leiaute corrente. Del Valle et al. (2012) utilizaram as mesmas abordagens de Del Valle (2010) e obtiveram resultados para um outro problema, o de corte de estoque bidimensional, no qual um conjunto de itens irregulares com uma demanda associada deve ser alocado na menor quantidade possível de placas retangulares. Mundim e Queiroz (2012) resolveram o 2PCP com um algoritmo híbrido, envolvendo GRASP e recozimento simulado. Os autores utilizaram a rotina de alocação de Del Valle et al. (2012) combinada com o recozimento simulado para melhorar a busca local do GRASP, com o objetivo de escapar de mínimos locais. Silveira (2013) apresentou duas novas heurísticas, baseadas em um algoritmo genético, para resolver o 2PCF. Essas heurísticas foram adaptadas para o corte de placas retangulares, uma vez que dependem basicamente da sequência em que os itens devem ser alocados. Como no corte de placas as dimensões do recipiente são fixas, quando um item da sequência não puder ser alocado, ele é descartado da solução. Dalalah et al. (2014) recentemente apresentaram uma abordagem para lidar com o corte de itens irregulares em placas retangulares e recipientes também irregulares. A estratégia é um algoritmo determinístico, baseado na união de polígonos, sempre visando a minimizar o desperdício da solução corrente.

Devido ao sucesso de meta-heurísticas na área de corte de itens irregulares, este trabalho estuda a utilização do algoritmo genético de chaves aleatórias viciadas (ou BRKGA, do inglês *Biased Random-Key Genetic Algorithm*) de acordo com a versão desenvolvida em Gonçalves e Resende (2011). Essa meta-heurística é um algoritmo genético, inspirado no algoritmo genético de chaves aleatórias (ou RKGA, do inglês *Random-Key Genetic Algorithm*) proposto por Bean (1994). Recentemente, Pinheiro et al. (2015) apresentaram uma abordagem RKGA para o 2PCF, competitiva com a literatura, baseada na técnica *bottom-left*. Além da diferença dos métodos, BRKGA e RKGA, a abordagem proposta neste trabalho utiliza outra técnica de codificação para o cromossomo e possui um espaço de soluções maior que Pinheiro et al. (2015), por usar uma heurística de decodificação do cromossomo que engloba a técnica *bottom-left*.

O método desenvolvido neste trabalho é baseado no BRKGA e parte do *framework* de Toso e Resende (2011) para resolver o 2PCF e 2PCP. Inicialmente, um total de P indivíduos são gerados. Cada indivíduo é um cromossomo de n posições, isto é, um vetor de n chaves. A partir de cada cromossomo é extraída a ordem em que os itens devem ser alocados no recipiente, observando o valor de cada chave no intervalo real [0, 1). Esse valor diz se o item é alocado no canto inferior esquerdo (valor maior do que 0,7) ou se o item é alocado no canto superior esquerdo do recipiente. A função de aptidão no 2PCF está relacionada ao comprimento da faixa, enquanto, no 2PCP, essa função observa a área ocupada do recipiente. Vale destacar que, exclusivamente no 2PCP, o item que não puder ser alocado no recipiente vai ser descartado do leiaute corrente. Após a criação do leiaute de cada indivíduo, separa-se o grupo com a população elite Pe, que contém os indivíduos mais aptos. Além disso, são gerados novos indivíduos de forma totalmente aleatória, formando a população dos mutantes Pm. Para completar a nova população de P indivíduos, filhos são gerados por meio da operação de cruzamento uniforme parametrizado envolvendo um pai elite e o outro selecionado do restante da população. O cruzamento consiste em pegar a chave do pai elite, considerando uma probabilidade viciada, ou do outro pai, até completar as n chaves. Esse procedimento é repetido a cada nova geração e continua até atingir um número máximo de iterações. A estratégia utilizada para evitar a sobreposição foi baseada em Toledo et al. (2013) e consiste em uma malha de pontos sobre a técnica de *no-fit polygon*.

O presente trabalho está organizado da seguinte forma. A Seção 2 apresenta a estratégia utilizada para evitar a sobreposição de itens. A Seção 3 detalha o BRKGA e apresenta o método de resolução proposto, assim como todas as rotinas implementadas. A Seção 4 discute os experimentos computacionais e a Seção 5 apresenta as conclusões e direções para trabalhos futuros.

2 CONCEITOS GEOMÉTRICOS

A geometria irregular dos itens é um desafio e diversas estratégias vêm sendo desenvolvidas para tratar dela. Um recente tutorial sobre as técnicas mais utilizadas é apresentado em Bennell e Oliveira (2008). Dessas estratégias, as mais utilizadas consistem no método *raster*, trigonometria direta, *phi function* e *no-fit polygon*.

A ideia do método *raster* é discretizar os itens em uma malha para verificar, por meio de uma soma de matrizes, a sobreposição. A desvantagem desta abordagem é que a qualidade da representação depende do refinamento da malha, ou seja, quanto mais refinada a discretização dos itens, mais próximo do item real e mais custosas são as operações sobre as suas representações. Em todas as outras abordagens, os itens são polígonos, o que deixa a representação geométrica precisa quando não se tem curvas. A Figura 1 apresenta dois itens ao lado esquerdo, um polígono convexo e o outro polígono não convexo. Esses polígonos são utilizados sem modificação na trigonometria direta e no caso da *phi function* e *no-fit polygon*. Por outro lado, no método *raster*, tem-se do lado direito da Figura 1 a discretização dos dois polígonos.

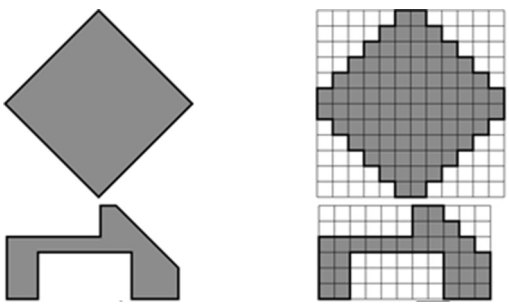

Figura 1 Exemplo de polígonos na trigonometria direta e no método *raster*.

A trigonometria direta utiliza os vértices e as arestas de um item para verificar sua sobreposição com outro item. Para isso, faz até quatro testes, a saber: (1) verifica a sobreposição da menor envoltória retangular dos itens e, se não existir sobreposição, os itens estão separados; (2) verifica a sobreposição entre todos os pares de arestas e, caso não exista sobreposição, os itens estão separados; (3) verifica se existe interseção entre as arestas dos itens e, se existir interseção, os itens estão sobrepostos; (4) verifica se cada um dos vértices de cada item está contido em outro item e, caso os vértices não estejam contidos, é garantia de que os itens estão separados.

As *phi functions* são expressões matemáticas que informam a posição relativa entre um par de itens quaisquer. A principal dificuldade desse método é a falta de um procedimento robusto para construir as expressões. Valores positivos indicam que os itens estão separados, valores negativos indicam que os itens estão sobrepostos e o valor zero indica que os itens estão se tocando.

O *no-fit polygon* é utilizado para verificar se pares de itens estão se tocando, separados ou sobrepostos. Essa técnica mantém todas as posições viáveis da trigonometria direta e reduz os cálculos a partir de uma etapa de pré-processamento, em que se faz o cálculo do *no-fit polygon* para todos os pares de itens e em todas as suas rotações. Em linhas gerais, um item é fixado no plano e o outro item (chamado de orbital) é transladado, dado um ponto de referência determinado, ao redor do item fixo. O caminho feito pelo ponto de referência do item orbital forma uma ou mais regiões (polígonos) do *no-fit polygon*.

Uma vez calculado o *no-fit polygon* entre dois itens, para verificar se há sobreposição entre eles, torna-se suficiente verificar se o ponto de referência do item orbital está fora de todas as regiões do *no-fit polygon* entre os itens. Se o ponto de referência estiver no interior de alguma região, os itens estão se sobrepondo. Se o ponto de referência estiver do lado de fora de todas as regiões, os itens estão separados. Se o ponto de referência estiver de fora das regiões e sobre os limites de pelo menos uma região, os itens estão encostados.

A abordagem utilizada neste trabalho é uma matriz baseada nos pontos do *no-fit polygon*. Se for usada apenas uma matriz com o método *raster*, a discretização de cada um dos itens é feita em separado e, assim, perde todos os encaixes não ortogonais. Para facilitar o entendimento, ver na Figura 2 uma solução obtida com o método *raster* da instância proposta por Oliveira et al. (2000) com quatro itens e uma placa de largura 12. O menor comprimento possível para alocar todos os itens nesse recipiente é 26.

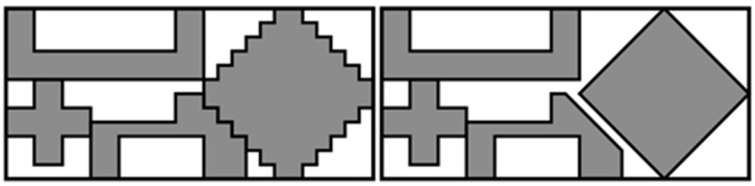

Figura 2 Solução obtida usando o método *raster* (esquerda) e representação poligonal da solução (direita).

Na Figura 2, nota-se que, usando o método *raster*, não se pode obter uma solução melhor. Entretanto, é claro que, ao considerar a representação poligonal dos itens, esse comprimento pode ser reduzido, deslocando o losango para próximo dos demais itens. Nesse caso, usando a abordagem deste trabalho, pode-se obter uma solução mais compacta, como a apresentada na Figura 3 A abordagem adotada neste trabalho é baseada em Toledo et al. (2013), em que define-se a malha de pontos sobre o *no-fit polygon*. Os itens continuam sendo representados por polígonos, mas o *no-fit polygon* é representado por uma malha. Para gerar os pontos da malha, foi implementado um método de triangulação simples, apresentado em O'Rourke (1998), e calculou-se o *no-fit polygon* parcial entre todos os

triângulos utilizando o algoritmo de ordenação de arestas de Cuninghame-Green (1989) sobre polígonos convexos.

Figura 3 Solução usando o método proposto neste trabalho.

3 MÉTODO DE RESOLUÇÃO

Os algoritmos genéticos foram inspirados em sistemas biológicos e utilizam conceitos de indivíduos (utilizados para representar as soluções, geralmente usando sequência de DNA ou cromossomo), hereditariedade (transmissão de características genéticas por meio da reprodução de indivíduos), mutação (indivíduos mutantes permitem que a população fuja de mínimos locais, explorando melhor o espaço de soluções), seleção natural (os melhores indivíduos se reproduzem e transmitem as características para as próximas gerações) e *crossover* (cruzamento entre indivíduos para a criação de novos descendentes). Esses algoritmos foram introduzidos por Holland (1975) e, desde então, diversos trabalhos e técnicas de otimização foram desenvolvidos a partir deles. Em linhas gerais, um algoritmo genético puro recebe uma lista de indivíduos e uma função de aptidão que calcula a qualidade de cada indivíduo. Enquanto algum critério de parada não for atingido, o algoritmo seleciona um conjunto de pais e faz a reprodução. Quando o critério de parada for satisfeito, o algoritmo retorna o melhor indivíduo da população.

O algoritmo genético com chaves aleatórias (ou RKGA, do inglês *Random-Key Genetic Algorithm*) proposto por Bean (1994) é uma melhoria do algoritmo genético puro. O RKGA representa cada indivíduo (ou solução do problema de otimização) como um cromossomo. Cada indivíduo é um vetor de chaves aleatórias, em que as chaves são números reais obtidos do intervalo contínuo [0, 1). Os indivíduos mais aptos, ou seja, os com melhor valor de função de aptidão têm mais chance de encontrar um parceiro e fazer a operação de cruzamento, podendo transmitir seu material genético para as próximas gerações. No RKGA, o cruzamento é feito combinando o material genético dos pais, com igual probabilidade para copiar tanto de um como do outro pai.

O algoritmo genético de chaves aleatórias viciadas, o BRKGA, carrega todos os princípios darwinistas do RKGA, como os indivíduos mais aptos. Todavia, durante o cruzamento no BRKGA, os indivíduos mais aptos têm uma probabilidade maior de transmitir o seu material genético para os seus sucessores. Com isso, pode-se dizer que durante o cruzamento existe um favorecimento dos pais com melhor valor de função de aptidão, permitindo gerar indivíduos de forma "viciada" (ou *biased*). Em outras palavras, a probabilidade de um filho herdar o material genético de um pai mais apto é estritamente maior do que 50%.

O algoritmo geral do BRKGA utilizado neste trabalho, denominado NestBRKGA, é baseado no modelo de Resende (2013), sem o critério de reinicialização. Cada indivíduo corresponde a um cromossomo de *n* posições, cada qual representando uma chave. As posições do cromossomo estão associadas à ordem em que os itens devem ser alocados no recipiente. Essa ideia surgiu no trabalho de Jakobs (1996), em que cada indivíduo é representado por um cromossomo, indicando a sequência de itens a serem alocados pela heurística *bottom-left*. Diferentemente do trabalho de Jakobs (1996), que procurava uma posição viável para cada item a partir do canto inferior esquerdo, procura-se uma posição viável a partir do canto inferior esquerdo ou do canto superior esquerdo.

A Tabela 1 traz os parâmetros do NestBRKGA seguindo os parâmetros pré-definidos no trabalho de Toso e Resende (2011). Em seguida, o Algoritmo 1 apresenta o pseudocódigo do algoritmo NestBRKGA.

Tabela 1 Parâmetros utilizados no NestBRKGA

Parâmetro	Descrição	Valores padrão
n	Número de elementos em cada cromossomo	Quantidade total de itens
P	Tamanho da população	1000
Pe	Tamanho da população elite	200
Pm	Número de mutantes em cada população	100
Pa	Probabilidade de cada chave ser herdada de um pai elite	70%

Fonte: Toso e Resende (2011).

No Algoritmo 1, o laço das linhas 2 a 26 representa a evolução de cada população (cada geração criada), repetindo enquanto o número de gerações for menor do que 100 e a solução não for ótima para o problema. Nesse problema, uma solução para o 2PCF é ótima se o comprimento for igual à soma da área dos itens dividido pela largura e, para o 2PCP, é ótima se todos os itens forem alocados.

Na linha 4 do Algoritmo 1, as soluções são avaliadas: retornando o comprimento para 2PCF e a área dos itens alocados para o 2PCP. Em seguida, na linha 5, é atribuído 20% dos melhores indivíduos para a população elite *Pe* e o restante para a polução não elite $P\,\bar{e}$. Os indivíduos mutantes são gerados na linha 6. É importante gerar os indivíduos mutantes de forma totalmente aleatória, pois são eles que permitem fugir de mínimos locais.

A operação de cruzamento é feita no laço das linhas 7 a 20 do Algoritmo 1. Nessa parte, destacam-se duas características importantes: (i) o cruzamento sempre é feito pegando um pai da população elite *Pe* e o outro da população não elite $P\,\bar{e}$; (ii) a probabilidade do material genético ser herdado do pai elite é igual a ρ_a (neste trabalho, foi usado 70%). Na linha 21, a população *P* recebe a nova população (da próxima geração) e, na linha 22, a melhor solução, conforme a função de aptidão do problema sendo resolvido, é encontrada e salva. Por fim, a melhor solução computada durante todas as gerações é retornada como resultado do algoritmo.

Algoritmo 1: NESTBRKGA

Entrada: P, P_e, P_m, n e ρ_a.
Saída: x^*, que é o melhor indivíduo da população.

```
1  início
2    enquanto critério de parada não for satisfeito faça
3      Gere a população de P indivíduos, cada um com n chaves aleatórias;
4      Avalie o custo de cada indivíduo de P;
5      Particione P em dois conjuntos: P_e e P_ē;
6      Adicione P_e, bem como P_m novos indivíduos mutantes à população P+ da próxima geração;
7      para i ← 1 até P − P_e − P_m faça
8        Escolha o pai a aleatoriamente de P_e;
9        Escolha o pai b aleatoriamente de P_ē;
10       para j ← 1 até n faça
11         Jogue uma moeda viciada com probabilidade ρ_a de dar cara;
12         se a moeda der cara então
13           c[j] ← a[j];
14         fim
15         senão
16           c[j] ← b[j];
17         fim
18       fim
19       Adicione o descendente c à população da próxima geração: P+ ← P+ ⋃ {c};
20     fim
21     Atualize a população: P ← P+;
22     Ache a melhor solução x+ em P;
23     se x+ é melhor do que x* então
24       x* ← x+;
25     fim
26   fim
27   retorna x*;
28 fim
```

3.1 Decodificador para o NestBRKGA

Como cada indivíduo é um vetor de chaves, é preciso decodificá-lo em forma de solução para o problema que está sendo resolvido. Para tanto, a heurística utilizada é uma heurística construtiva, baseada na heurística *bottom left*.

Devido à grande quantidade de trabalhos que utilizam a heurística *bottom left*, principalmente por obter boas soluções em um tempo computacional muito baixo, a heurística proposta neste trabalho aloca os itens o mais à esquerda possível, combinando a escolha do canto inferior esquerdo à do canto superior esquerdo. Assim, a diferença dessa heurística para a heurística *bottom left* tradicional é que se pode alocar no canto inferior esquerdo (*bottom left*) ou no superior esquerdo (*top left*). A estratégia, denominada aqui *bottom-top left*, é apresentada no Algoritmo 2.

Algoritmo 2: *Bottom-top left*

Entrada: $Item, Chave, n, L$ e C.
$Item$ é o vetor de itens.
$Chave$ é o vetor de chaves aleatórias associadas aos itens.
n é a quantidade de itens.
L é a largura do recipiente.
C é o comprimento do recipiente.
Saída: S
S é solução com os itens alocados no recipiente.

```
início
    //Alocando o primeiro item do vetor de itens na posição (0,0) do recipiente.
    S ← {Item[1], (0,0)};
    recipiente ← {Item[1], (0,0)};
    para i ← 2 até n faça
        se Chave[i] < 0,7 então
            //Procurando um ponto factível pela heurística bottom left para alocar o item i no recipiente.
            para x ← 0 até C faça
                para y ← 0 até L faça
                    se {Item[i], (x, y)} pode ser alocado no recipiente então
                        S ← {S} ⋃ {Item[i], (x, y)};
                        recipiente ← {recipiente} ⋃ {Item[i], (x, y)};
                        //Sai dos laços.
                        x ← C + 1; y ← L + 1;
                    fim
                fim
            fim
        fim
        senão
            //Procurando um ponto factível pela heurística top left para alocar o item i no recipiente.
            para x ← 0 até C faça
                para y ← L até 0 faça
                    se {Item[i], (x, y)} pode ser alocado no recipiente então
                        S ← {S} ⋃ {Item[i], (x, y)};
                        recipiente ← {recipiente} ⋃ {Item[i], (x, y)};
                        //Sai dos laços.
                        x ← C + 1; y ← L + 1;
                    fim
                fim
            fim
        fim
    fim
fim
retorna x*;
```

Basicamente, o Algoritmo 2 recebe do NestBRKGA os itens ordenados, o vetor de chaves aleatórias e as dimensões do recipiente. Aloca o primeiro item no ponto inferior esquerdo. Em seguida, para os demais itens, ele aloca pela *bottom*

left, se o valor da chave for menor do que 70%, e pela *top left*, caso contrário. Observa-se que, nas linhas 8 e 9, a heurística *bottom left* busca o ponto de alocação observando de forma crescente o eixo *y* e, em seguida, o eixo *x* até atingir os limites do recipiente (largura e comprimento). Por outro lado, as linhas 21 e 22, da heurística *top left* faz a busca observando os valores em ordem decrescente pelo eixo *y* e crescente pelo eixo *x*.

O Algoritmo 2 é utilizado no decodificador do NestBRKGA. Para cada indivíduo, pode-se gerar um leiaute viável, respeitando a ordem dos itens no cromossomo e o valor das chaves associadas, de tal forma a não alocar os itens que excedem as dimensões do recipiente e causam sobreposição.

Para o nosso exemplo, considere um recipiente de largura 30, comprimento 14 e quatro itens. Os itens são identificados por cores e é associado um identificador para cada um, a saber: 1, 2, 3 e 4. Considere o cromossomo de um indivíduo dado por [3 1 4 2]. Esse cromossomo informa que o primeiro item a ser alocado é o item 3, seguido pelos itens 1, 4 e 2, respectivamente. Além do cromossomo com a ordem, também é dado o vetor de chaves aleatórias [0,8 0,3 0,9 0,8].

A Figura 4 ilustra a solução do nosso exemplo. Observe que apenas o item 1 (segundo item do cromossomo) foi alocado na parte inferior, enquanto os outros três foram alocados pela heurística *top left* de acordo com valor do parâmetro ρ_e. É importante lembrar que, no caso do 2PCP, existem itens que podem ficar de fora da solução, enquanto que, no 2PCF, todos os itens devem ser alocados.

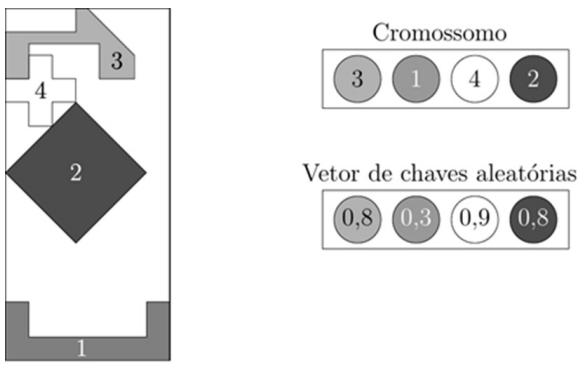

Figura 4 Solução do exemplo obtida pelo NestBRKGA.

O presente trabalho utiliza o *framework* (<http://mauricio.resende.info/src/brkgaAPI/>) do BRKGA proposto por Toso e Resende (2011), que contém um conjunto de rotinas codificadas na linguagem de programação em C++.

4 EXPERIMENTOS COMPUTACIONAIS

Os algoritmos utilizados no presente trabalho foram implementados na linguagem C++, seguindo o *framework* de Toso e Resende (2011) e, assim, viabilizou-se a comunicação entre as rotinas utilizadas e o que já estava implementado. Todos os experimentos foram feitos em um computador com processador Intel Core i7-2600, com processador de 3.4 GHz e 16 GB de memória RAM, no sistema operacional Linux Ubuntu 14.04.

Esta seção foi dividida em três partes. A primeira traz os experimentos computacionais para o 2PCF. A segunda apresenta os resultados obtidos para o 2PCP. Por fim, a terceira parte traz os experimentos computacionais do NestBRKGA frente ao RKGA proposto por Pinheiro et al. (2015). Para uma comparação justa, algumas modificações nos parâmetros são apresentadas. Devido à aleatoriedade presente no NestBRKGA, ele foi executado 10 vezes para resolver cada uma das instâncias.

4.1 Resultados para o 2PCF

Foram utilizadas 23 instâncias da literatura para efeitos de comparação dos resultados obtidos. Devido à estratégia de sobreposição extraída de Toledo et al. (2013), nosso método possui uma limitação associada à discretização da malha de pontos. Nesse sentido, para verificar se o nosso método consegue encontrar boas soluções usando a malha considerada, usou-se instâncias de Toledo et al. (2013) e outras testadas por Fischetti e Luzzi (2009) e Alvarez-Valdes et al. (2013).

As instâncias utilizadas são apresentadas na Tabela 2. As instâncias RCO*i*, BLAZEWICZ*i*, SHAPES*j*, SHAPES9 e SHAPES15 são variações de instâncias clássicas de mesmo nome (mais detalhes são dados em Toledo et al. (2013)). As demais instâncias já foram utilizadas em outros trabalhos da literatura (GOMES; OLIVEIRA, 2006). Vale destacar que para todas as instâncias não foi permitido rotacionar os itens. A Tabela 2 traz o nome da instância, seguido pela quantidade de itens, largura da faixa e onde a instância pode ser encontrada.

A Tabela 3 reporta os resultados para as variações das instâncias RCO, BLAZEWICZ e SHAPES propostas em Toledo et al. (2013). Algumas dessas instâncias possuem solução ótima para a malha de pontos usada. Na segunda coluna da tabela, as soluções ótimas são apresentadas como Limitantes Inferiores (LI). As colunas seguintes reportam o comprimento encontrado, a ocupação e o tempo computacional em segundos. Cada coluna apresenta o melhor resultado, a média aritmética obtida sobre as 10 execuções do método e o pior resultado encontrado. Para as instâncias com dois asteriscos, a solução ótima foi encontrada em todas as execuções. Para a instância BLAZEWICZ2, marcada com um asterisco apenas, a solução ótima foi computada em 8 das 10 iterações.

Tabela 2 Informações sobre as instâncias para o 2PCF

Instância	Número de itens	Largura da faixa	Usada em
RCO*i*	7*i* para *i* = 1,..,5.	15	Toledo et al. (2013)
BLAZEWICZ*i*	7*i* para *i* = 1,..,5.	15	Toledo et al. (2013)
SHAPES*j*	4*j* para *j* = 2, 4 e 7.	40	Toledo et al. (2013)
SHAPES9	34	40	Toledo et al. (2013)
SHAPES15	43	40	Toledo et al. (2013)
GLASS1	5	45	Fischetti; Luzzi (2009)
GLASS2	7	45	Fischetti; Luzzi (2009)
GLASS3	9	100	Fischetti; Luzzi (2009)
DIGHE1	16	100	Alvarez-Valdes et al. (2013)
DIGHE2	10	100	Alvarez-Valdes et al. (2013)
FU	12	38	Alvarez-Valdes et al. (2013)
JAKOBS1	25	40	Alvarez-Valdes et al. (2013)
JAKOBS2	25	70	Alvarez-Valdes et al. (2013)

No pior caso, o tempo computacional apresentado na Tabela 3 foi de aproximadamente 100 segundos para a SHAPES15 (também conhecida na literatura como shapes0). Com esse tempo, o método poderia ser utilizado para gerar uma solução inicial para o modelo proposto em Toledo et al. (2013), tendo em vista que, nesse modelo, os autores utilizaram soluções iniciais devido à dificuldade em encontrar soluções factíveis para as instâncias maiores, como SHAPES9 e SHAPES15. As células em branco da tabela indicam que todos os valores da coluna foram iguais, sendo reportado apenas o valor médio. A Figura 5 apresenta algumas das soluções ótimas, em especial, aquelas encontradas para as maiores instâncias.

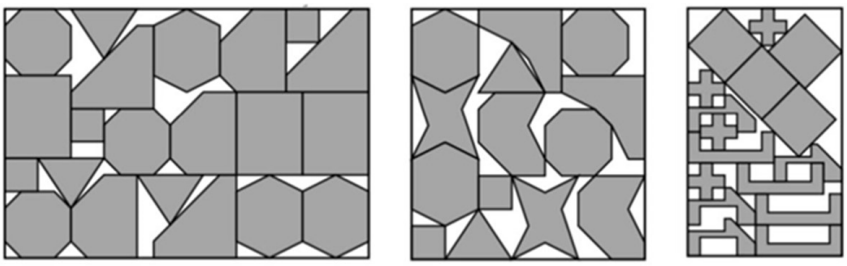

Figura 5 Soluções ótimas para algumas das instâncias na Tabela 3.

Recentemente, MirHassani e Bashirzadeh (2015) propõem um conjunto de meta-heurísticas GRASP que também utiliza a malha de pontos proposta em

Tabela 3 Resultado para o 2PCF usando as instâncias de Toledo et al. (2013)

Instância	LI	Comprimento			Ocupação			Tempo (s)		
		Melhor	Média	Pior	Melhor	Média	Pior	Melhor	Média	Pior
RCO1**	8		8			0,7833		2,6	2,7	2,9
RCO2**	15		15			0,8356		6,6	6,7	6,9
RCO3**	22		22			0,8545		11,5	11,7	12,0
RCO4		29	29,9	30	0,8643	0,8384	0,8356	16,7	17,0	17,7
RCO5			37			0,8468		23,3	23,9	24,6
BLAZEWICZ1**	8		8			0,6750		2,6	2,7	2,8
BLAZEWICZ2*	14	14	14,2	15	0,7714	0,7611	0,7200	6,5	6,7	6,9
BLAZEWICZ3	20		21			0,7714		11,5	11,7	11,9
BLAZEWICZ4			28			0,7714		16,6	17,2	17,7
BLAZEWICZ5			35			0,7714		23,3	23,9	24,6
SHAPES2**	14		14			0,5714		9,4	9,6	9,9
SHAPES4**	25		25			0,6400		21,2	22,0	22,6
SHAPES7			42			0,6667		49,3	50,0	51,2
SHAPES9		48	48,7	49	0,6750	0,6654	0,6612	65,4	66,2	68,3
SHAPES15		61	61,8	62	0,6540	0,6457	0,6435	95,8	97,4	99,3

Toledo et al. (2013). A Tabela 4 apresenta um resumo dos melhores resultados obtidos em MirHassani e Bashirzadeh (2015) e pelo NestBRKGA. Essa tabela apresenta, para cada instância, o menor comprimento obtido e o tempo computacional médio de gasto. O computador utilizado por MirHassani e Bashirzadeh (2015) é um computador com 4 GB de memória RAM e um processador com 2.2 GHz, inferior ao computador utilizado no presente trabalho, por isso evita-se a comparação de tempo. Os valores com um asterisco indicam que o comprimento obtido é ótimo para a malha de pontos utilizada. Ambos os métodos obtiveram o ótimo para sete instâncias. O NestBRKGA foi melhor ou igual em 12 instâncias analisadas, obtendo resultados estritamente melhores para cinco instâncias (RCO5, BLAZEWICZ4, SHAPES4, SHAPES7 e SHAPES9) e resultados inferiores para três instâncias (RCO4, BLAZEWICZ3 e SHAPES15).

Tabela 4 Resultado para o 2PCF comparado com a literatura

Instância	MirHassani e Bashirzadeh (2015)		NestBRKGA	
	Comprimento	Tempo (s)	Comprimento	Tempo (s)
RCO1	8*	1,0	8*	2,7
RCO2	15*	5,7	15*	6,7
RCO3	22*	14,5	22*	11,7
RCO4	27	28,3	29	17,0
RCO5	38	43,9	37	23,7
BLAZEWICZ1	8*	0,9	8*	2,7
BLAZEWICZ2	14*	4,8	14*	6,7
BLAZEWICZ3	20*	13,8	21	11,7
BLAZEWICZ4	30	28,0	28	17,2
BLAZEWICZ5	35	40,3	35	23,9
SHAPES2	14*	3,9	14*	9,6
SHAPES4	27	42,0	25*	22,0
SHAPES7	47	240,6	42	50,0
SHAPES9	55	301,7	48	66,2
SHAPES15	57	311,1	61	97,4

A Tabela 5 reporta o resultado para as instâncias GLASS1, GLASS2, GLASS3, DIGHE1 e DIGHE2, que são de encaixe perfeito e foram construídas a partir do corte de quadrados de lado 45 e 100, e para FU, JAKOBS1 e JAKOBS2. A estrutura da Tabela 5 é idêntica à estrutura da Tabela 3. Em geral, o tempo computacional continua baixo, ultrapassando o tempo de cinco minutos apenas para a instância DIGHE2.

Tabela 5 Resultado para o 2PCF considerando as demais instâncias

Instância	LI	Comprimento			Ocupação			Tempo (s)		
		Melhor	Média	Pior	Melhor	Média	Pior	Melhor	Média	Pior
GLASS1**	45	45			1,0000			37,2	37,8	38,7
GLASS2**	45	45			1,0000			1,5	1,5	1,5
GLASS3**	100	100			1,0000			8,9	14,5	21,0
DIGHE1**	100	100			1,0000			56,8	92,4	113,5
DIGHE2	100	112			0,8928			369,8	381,8	385,2
FU		34			0,8382			48,1	44,5	50,3
JAKOBS1		12			0,8167			34,2	35,2	36,1
JAKOBS2		26			0,7423			132,3	137,7	144,2

Observando os resultados na Tabela 5, o encaixe perfeito foi obtido para quatro instâncias atingindo 100% de ocupação em todas as 10 execuções, para as instâncias GLASS1, GLASS2, GLASS3 e DIGHE1. O desenho das soluções é apresentado na Figura 6.

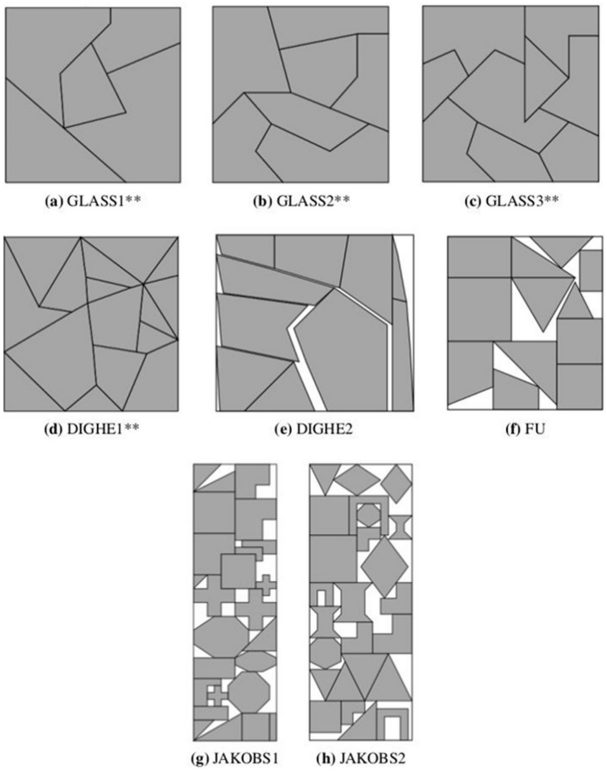

Figura 6 Soluções encontradas para as instâncias na Tabela 5.

4.2 Resultados para o 2PCP

Para verificar o desempenho do método sobre o 2PCP, os experimentos foram separados em dois grupos. O primeiro grupo contém cinco instâncias usadas na literatura e que não foram testadas na Seção 3, os resultados foram comparados com os de Del Valle et al. (2012) e Silveira (2013). O segundo grupo apresenta os resultados que são comparados com os experimentos de Dalalah et al. (2014) que possuem recipiente retangular. A Tabela 6 apresenta os dados das instâncias utilizadas nesta seção. Ela reporta o nome da instância, a quantidade de itens, a largura e o comprimento do recipiente e, por último, as rotações permitidas para cada instância.

Tabela 6 Informações sobre as instâncias para o 2PCP

Instância	Número de itens	Largura da placa	Comprimento da placa	Rotações permitidas
FU	12	38	34,00	0; 90; 180; 270
SHAPES1	43	40	59,00	0; 180
MARQUES	24	104	83,60	0; 90; 180; 270
SHIRTS	99	40	63,13	0; 180
TROUSERS	64	79	245,75	0; 180
EXEMPLO 2	13	10	6,00	0; 90; 180; 270
EXEMPLO 3	16	17	17,00	0

A Tabela 7 apresenta os resultados de Del Valle et al. (2012), os obtidos pelas duas heurísticas de Silveira (2013) e os computados pelo método aqui proposto, o NestBRKGA. Ela apresenta, para cada algoritmo, a ocupação da placa e o tempo computacional gasto. As melhores taxas de ocupação foram marcadas em negrito. Os dois asteriscos em frente ao nome das instâncias indicam que o nosso algoritmo conseguiu encontrar a solução ótima em todas as execuções do método, ou seja, o método conseguiu alocar todos os itens na placa em todas as execuções. O tempo computacional reportado para o nosso método foi tomado como a média das 10 execuções.

Observando a Tabela 7, NestBRKGA obtém a solução ótima para todas as rodadas das instâncias FU, SHAPES1, MARQUES e SHIRTS. A instância TROUSERS apresentou uma solução bem próxima da melhor solução da literatura, com uma diferença menor que 0,0098%. Vale mencionar que não foi feita uma comparação direta do tempo computacional, pois os métodos foram implementados em máquinas diferentes. Sendo assim, uma comparação seria injusta e o tempo é apresentado por completude. Em geral, o tempo do algoritmo proposto é pequeno para todas as instâncias, embora seja superior para a SHIRTS e a TROUSERS. Isso ocorre naturalmente, pois a quantidade de pontos utilizados para representar a não sobreposição é muito grande comparado com as demais instâncias.

Tabela 7 Resultados para o 2PCP e comparação com a literatura

Instância	Del Valle et al. (2012)		H1- Silveira (2013)		H2- Silveira (2013)		NestBRKGA	
	Ocupação	Tempo	Ocupação	Tempo	Ocupação	Tempo	Ocupação	Tempo
FU	0,8382	21,8	0,8382	13,2	0,8382	69,7	0,8382	3,0
SHAPES1	0,6424	3891,6	0,6559	1202,0	0,6559	1203,1	0,6763	2,8
MARQUES	0,8274	217,8	0,8274	58,2	0,8274	63,2	0,8274	27,3
SHIRTS	0,7702	14317,1	0,8471	1207,5	0,8389	1220,1	0,8553	135,1
TROUSERS	0,7866	5796,6	0,8745	1202,0	0,8830	1206,4	0,8732	237,3

O trabalho de Dalalah et al. (2014) apresenta seis experimentos computacionais para o 2PCP. Porém, apenas os experimentos números dois e três utilizam uma placa retangular. O resultado para essas duas instâncias está na Tabela 8. O desempenho do NestBRKGA, comparado com o algoritmo de Dalalah et al. (2014), foi superior para o EXEMPLO 2 e igual para o EXEMPLO 3, que neste caso é a solução ótima. As melhores taxas de ocupação estão em negrito, destacando que a mesma taxa de ocupação foi obtida em todas as 10 execuções. As soluções para as instâncias das Tabelas 7 e 8 estão apresentadas na Figura 7.

Tabela 8 Comparação com os experimentos de Dalalah et al. (2014)

Instância	Dalalah et al. (2014)		NestBRKGA					
	Ocupação	Tempo	Ocupação			Tempo		
			Melhor	Média	Pior	Melhor	Média	Pior
EXEMPLO 2	0,9000	11,5		0,9500		6,1	6,2	6,4
EXEMPLO 3	0,8858	12,4		0,8858		0,1	0,1	0,1

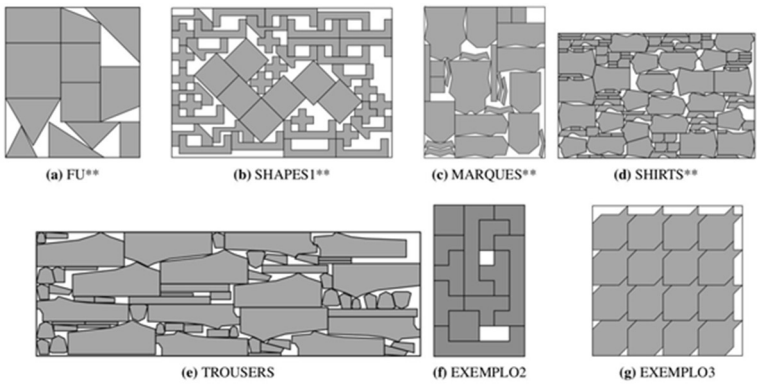

Figura 7 Soluções encontradas pelo NestBRKGA para o 2PCP.

4.3 Resultados comparado com o RKGA

Pinheiro et al. (2015) apresentaram um algoritmo RKGA para problemas de corte de itens irregulares. Esse algoritmo é semelhante ao NestBRKGA. As duas abordagens se diferenciam na forma como o cruzamento é feito. Além disso, em Pinheiro et al. (2015), os autores codificam o cromossomo para fornecer a sequência dos itens a serem alocados por uma técnica de alocação *bottom left*, já o NestBRKGA combina o canto inferior esquerdo (*bottom left*) e canto superior esquerdo (*top left*) para atingir boas soluções. Nesse trabalho, outro diferencial do NestBRKGA é o conceito de chaves viciadas, o que segundo Resende (2013) pode acelerar a convergência do método.

A Tabela 9 apresenta os parâmetros utilizados por Pinheiro et al. (2015). Para fazer uma comparação com os autores utilizou-se, nesta seção, estes parâmetros para o NestBRKGA. O número de gerações utilizados por Pinheiro et al. (2015) e no NestBRKGA, nesta seção, é de 200 gerações.

Tabela 9 Parâmetros utilizados por Pinheiro et al. (2015)

Parâmetro	Descrição	Valores Padrão
P	Tamanho da população	800
Pe	Tamanho da população elite	160
Pm	Número de mutantes em cada iteração	400

Fonte: traduzida de Pinheiro et al. (2015).

A Tabela 10 apresenta as dez instâncias testadas por Pinheiro et al. (2015) para resolver o 2PCP. Note que a largura da placa das instâncias ALBANO (4900) e SWIM (5752) é muito superior às demais instâncias. Nesse caso, devido ao elevado custo de memória usado para representar as malhas de pontos dessas duas instâncias, o NestBRKGA atingiu a capacidade de memória do computador utilizado, não conseguindo obter soluções factíveis. Isso ocorre pois, neste trabalho, foi implementada uma malha discreta que utiliza uma discretização de 1 para 1, dos itens e dos recipientes.

A Tabela 11 mostra a competitividade do NestBRKGA e do RKGA de Pinheiro et al. (2015). Das oito instâncias para as quais o NestBRKGA encontrou solução factível, obteve a melhor solução ótima para duas instâncias, empatou em duas e perdeu em quatro. Com relação à ocupação média das dez iterações dos métodos, obteve uma solução média melhor para cinco dos dez experimentos. Com relação ao tempo computacional, os autores do RKGA usaram um computador com processador Intel i5 com 4 GB de memória RAM, inferior ao computador usado no presente trabalho.

Tabela 10 Informações sobre as instâncias de Pinheiro et al. (2015)

Instância	Número de itens	Largura da placa	Rotações permitidas
ALBANO	24	4900	0; 180
DIGHE1	16	100	0
DIGHE2	10	100	0
JAKOBS1	25	40	0; 90; 180; 270
JAKOBS2	25	70	0; 90; 180; 270
MARQUES	24	104	0; 90; 180; 270
SHAPES0	43	40	0
SHAPES1	43	40	0; 180
SWIM	48	5752	0; 180
TROUSERS	64	79	0; 180

Fonte: traduzida de Pinheiro et al. (2015).

Tabela 11 Resultados para o NestBRKGA comparado com o RKGA

Instância	RKGA de Pinheiro et al. (2015)				NestBRKGA			
	Ocupação			Tempo (s)	Ocupação			Tempo (s)
	Melhor	Média	Pior		Melhor	Média	Pior	
ALBANO	0,8474	0,8469	0,8462	10230,0				
DIGHE1	1,0000	0,9996	0,9990	2053,0	1,0000	1,0000	1,0000	183,1
DIGHE2	1,0000	0,9999	0,9952	2291,0	0,8918	0,8918	0,8918	636,9
JAKOBS1	0,8167	0,7879	0,7661	10268,0	0,8167	0,8167	0,8167	114,8
JAKOBS2	0,7975	0,7790	0,7530	12205,0	0,7720	0,7482	0,7423	663,4
MARQUES	0,8447	0,8431	0,8418	6871,0	0,8756	0,8723	0,8647	2386,1
SHAPES0	0,6725	0,6350	0,6075	1826,0	0,6557	0,6420	0,6349	159,2
SHAPES1	0,6637	0,6598	0,6342	3025,0	0,7143	0,6932	0,6877	322,9
SWIM	0,7359	0,7078	0,6844	14352,0				
TROUSERS	0,9091	0,8867	0,8637	16025,0	0,8677	0,8584	0,8508	3612,3

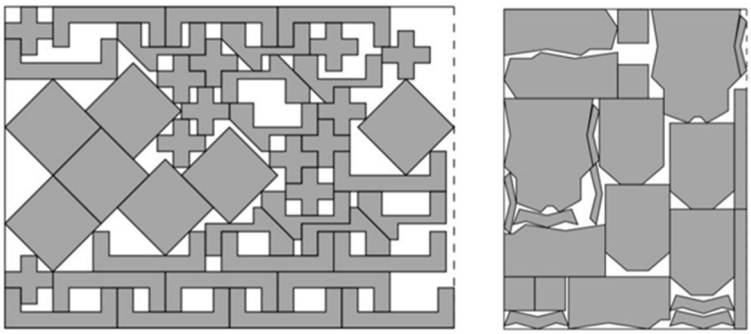

Figura 8 Algumas soluções encontradas pelo NestBRKGA.

5 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho apresenta uma abordagem baseada no algoritmo genético de chaves aleatórias viciadas para dois problemas de corte de itens irregulares. O primeiro problema, denominado 2PCF, busca minimizar o comprimento do recipiente usado, enquanto o outro, chamado de 2PCP, tem o objetivo de maximizar a ocupação da área do recipiente. Para a resolução desses problemas, foi implementada a técnica de verificação de não sobreposição de Toledo et al. (2013), na qual o *no-fit polygon* é discretizado em uma malha de pontos. Para a implementação do algoritmo genético, foi utilizado o *framework* de Toso e Resende (2011).

O *framework* é muito dependente do algoritmo de decodificação utilizado. Neste trabalho, foi aplicado um algoritmo simples para decodificar os indivíduos. Cada indivíduo é formado por um cromossomo, que representa a sequência em que os itens serão alocados no recipiente e um vetor com as chaves viciadas, que se refere à estratégia utilizada para alocar cada item. Foram utilizadas as estratégias do canto inferior esquerdo (*bottom left*) e do canto superior esquerdo (*top left*).

O algoritmo proposto foi testado e comparado, para cada um dos problemas, com trabalhos recentes da literatura especializada. No 2PCF, foram consideradas 23 instâncias, para as quais 11 tiveram a solução ótima encontrada conforme a malha de pontos utilizada para evitar a sobreposição. Das 23 instâncias analisadas, foram encontrados métodos heurísticos na literatura para 15 instâncias comparados aos quais o método proposto: melhorou em cinco, perdeu em três e empatou nas outras sete. Para o 2PCP, foram consideradas sete instâncias. Desse conjunto, cinco instâncias foram resolvidas na otimalidade (todos os itens foram alocados no recipiente). Embora não tenha resolvido duas das instâncias na otimalidade, o método consegue melhorar a solução de uma instância, apresentando uma taxa de ocupação de 95% contra um aproveitamento de 90% da literatura.

Além desses experimentos, o algoritmo proposto foi comparado com um recente algoritmo genético de chaves aleatórias proposto na literatura. Nesses experimentos, dos dez computacionais, nosso método obteve soluções médias melhores para 50% dos experimentos e melhorou dois resultados publicados.

A partir dos experimentos computacionais realizados, pode-se concluir que o método proposto foi aplicado com sucesso na área de corte de itens irregulares, sendo bastante competitivo com a literatura e melhorando resultados recentes. Note que a estratégia proposta é facilmente adaptável para outros problemas de corte e empacotamento, podendo ser usada como ponto de partida para outros trabalhos de pesquisa.

Como trabalhos futuros, pretende-se estender o algoritmo para recipientes irregulares e testar novos algoritmos de decodificação que não sejam dependentes da malha de pontos. Outros objetivos também incluem considerar a resolução de outros problemas, como o *bin packing* e o *cutting stock*.

AGRADECIMENTOS

Este trabalho contou com o apoio financeiro da CAPES, CNPq (processos 476792/2013-4 e 471351/2012-1), FAPESP (processos 2010/10133-0 e 2013/07375-0) e FAPEG.

REFERÊNCIAS

ALVAREZ-VALDES, R.; MARTINEZ, A.; TAMARIT, J. A branch and bound algorithm for cutting and packing irregularly shaped pieces. **International Journal of Production Economics**, v. 145, n. 2, p. 463-477, 2013.

ALVES, C. et al. New constructive algorithms for leather nesting in the automotive industry. **Computers & Operations Research**, v. 39, n. 7, p. 1487-1505, 2012.

BALDACCI, R. et al. Algorithms for nesting with defects. **Discrete Applied Mathematics**, v. 163, p. 17-33, 2014.

BEAN, J. C. Genetic algorithms and random keys for sequencing and optimization. **ORSA journal on computing**, v. 6, n. 2, p. 154-160, 1994.

BENNELL, J. A.; OLIVEIRA, J. F. The geometry of nesting problems: a tutorial. **European Journal of Operational Research**, n. 184, p. 397-415, 2008.

BENNELL, J. A.; SONG, X. A. A beam search implementation for the irregular shape packing problem. **Journal of Heuristics**, n. 16, p. 167-188, 2010.

BURKE, E. et al. A new bottom-left-fill heuristic algorithm for the two-dimensional irregular packing problem. **Operations research**, v. 54, n. 3, p. 587-601, 2006.

CARRAVILLA, M. A.; RIBEIRO, C.; OLIVEIRA, J. F. Solving nesting problems with non-convex polygons by constraint logic programming. **International Transactions in Operational Research**, v. 10, n. 6, p. 651-663, 2003.

CUNINGHAME-GREEN, R. Geometry, shoemaking and the milk tray problem. **New Scientist**, 123, p. 6-20, 1989.

DALALAH, D.; KHRAIS, S.; BATAINEH, K. Waste minimization in irregular stock cutting. **Journal of Manufacturing Systems**, v. 33, p. 27-40, 2014.

DEL VALLE, A. M. **Problema da mochila com itens irregulares.** Dissertação (Mestrado) – Universidade Estadual de Campinas, Campinas, 2010.

DEL VALLE, A. M. et al. Heuristics for two-dimensional knapsack and cutting stock problems with items of irregular shape. **Expert Systems with Applications**, v. 39, p. p. 12589-12598, 2012.

DOWSLAND, K. A.; VAID, S.; DOWSLAND, W. B. An algorithm for polygon placement using a bottom-left strategy. **European Journal of Operational Research**, v. 141, p. 371-381, 2002.

ELKERAN, A. A new approach for sheet nesting problem using guided cuckoo search and pairwise clustering. **European Journal of Operational Research**, v. 231, p. 757-769, 2013.

FISCHETTI, M.; LUZZI, I. Mixed-integer programming models for nesting problems. **Journal of Heuristics**, v. 15, n. 3, p. 201-226, 2009.

GAREY, M. R.; JOHNSON, D. S. **Computers and intractability: a guide to the theory of np-hardness.** San Francisco: Freeman, 1979. p. 124-127.

GOMES, A. M.; OLIVEIRA, J. F. A 2-exchange heuristic for nesting problems. **European Journal of Operational Research**, v. 141, p. 359-370, 2002.

_____. Solving irregular strip packing problems by hybridising simulated annealing and linear programming. **Journal of the Operational Research Society**, v. 171, p. 811-829, 2006.

GONÇALVES, J. F.; RESENDE, M. G. Biased random-key genetic algorithms for combinatorial optimization. **Journal of Heuristics**, v. 17, n. 5, 487-525, 2011.

HOLLAND, J. H. **Adaptation in natural and artificial systems.** University of Michigan Press, Michigan, 1975.

IMAMICHI, T.; YAGIURA, M.; NAGAMOCHI, H. An iterated local search algorithm based on nonlinear programming for the irregular strip packing problem. **Discrete Optimization**, v. 6, p. 345-361, 2009.

JAKOBS, S. On genetic algorithms for the packing of polygons. **European Journal of Operational Research**, v. 88, p. 165-18, 1996.

LEÃO, A. A. S. et al. A semi-continuous MIP model for the irregular strip packing problem. **International Journal of Production Research**, p. 37-41, 2015.

LEE, C. S. et al. Ship part nesting by pattern recognition and group arrangement. **Robotics and Computer-Integrated Manufacturing**, p. 56-63, 2013.

MIRHASSANI, S. A.; BASHIRZADEH, A. J. A GRASP meta-heuristic for two-dimensional irregular cutting stock problem. **The International Journal of Advanced Manufacturing Technology**, p. 1-10, 2015.

MUNDIM, L. R.; QUEIROZ, T. A. A hybrid heuristic for the 0-1 knapsack problem with items of irregular shape. In: **Informática (CLEI), 2012 XXXVIII Conferência Latino-americana**, p. 1-6, 2012.

NIELSEN, B.; ODGAARD, A. Fast neighborhood search for the nesting problem. **Relatório técnico, DIKU**, p. 9-10, 2003.

OLIVEIRA, J. F.; GOMES, A. M.; FERREIRA, J. S. Topos: a new constructive algorithm for nesting problems. **OR Spectrum**, v. 22, p. 263-284, 2000.

O'ROURKE, J. **Computational Geometry in C.** Cambridge, 1998. p.33-43.

PINHEIRO, P. R.; JÚNIOR, B. A.; SARAIVA, R. D. A random-key genetic algorithm for solving the nesting problem. **International Journal of Computer Integrated Manufacturing**, p. 1-7, 2015.

RESENDE, M. G. C. Introdução aos algoritmos genéticos de chaves aleatórias viciadas. **Simpósio Brasileiro de Pesquisa Operacional**, p. 3680-3691.

SILVEIRA, T. **Problemas de empacotamento com itens irregulares: Heurísticas e avaliação de construtores de NFP.** Dissertação (Mestrado) – Universidade Estadual de Campinas, Campinas, 2013.

SPEARS, W. M.; DEJONG, K. A. On the virtues of parameterized uniform crossover. **Proceedings of the Fourth International Conference on Genetic Algorithms**, p. 230-236, 1991.

TOLEDO, F. M. et al. The dotted-board model: a new MIP model for nesting irregular shapes. **International Journal of Production Economics**, v. 145 n. 2, p. 478-487, 2013.

TOSO, R. F.; RESENDE M. G. C. A C++ application programming interface for bised random-key genetic algorithms. **ATandT Labs Research Technical Report.**

UMETANI, S. et al . Solving the irregular strip packing problem via guided local search for overlap minimization. **International Transactions in Operational Research**, v. 16, n. 6, p. 661-683, 2009.

YANG, H. H.; LIN, C.L. On genetic algorithms for shoe making nesting: a Taiwan case. **Expert Systems with Applications**, p. 1134-1141, 2009.

YANG, X. S.; DEB, S. Cuckoo search via Lévy flights. **World congress on nature and biologically inspired computing (NaBIC)**, p. 210-214, 2010.

WASCHER, G.; HAUBNER, H.; SCHUMANN, H. 2007. An improved typology of cutting and packing problems. **European Journal of Operational Research**, v. 183, p. 1109-1130.