Proposta de método construtivo para o problema de posicionamento de formas irregulares

Alexandre Romanelli[1], Mauro N. Rocha[1]

Departamento de Informática – Universidade Federal de Viçosa (UFV)
Caixa Postal 15.064 – 91.501-970 – Viçosa – MG – Brazil
{alexandre.romanelli,mnacif}@ufv.br

Resumo O problema de posicionamento de formas irregulares bidimensionais é abordado neste artigo, que apresenta um novo algoritmo construtivo para obter soluções válidas. Os trabalhos relacionados que foram consultados apresentam métodos exatos, que determinam solução ótima, e métodos heurísticos que são comumente preferíveis por consequência da complexidade do problema. Nestes, há uma preocupação frequente em construir boas soluções iniciais para os métodos de busca propostos. O algoritmo descrito neste trabalho constitui uma inovação por ser baseado na intercalação de posicionamentos de itens e operações de compactação com programação linear. Os experimentos realizados indicaram que este método é competitivo, tendo atingido, em 12 das 14 instâncias testadas, resultados melhores do que os algoritmos construtivos para este problema que foram encontrados na literatura. Este artigo descreve o algoritmo básico proposto, as heurísticas para seleção de próximo item e as regras de posicionamento utilizadas nos experimentos, além do método de compactação e das estratégias de formação de intervalos entre as compactações. Os resultados obtidos com os experimentos são analisados e comparados aos melhores resultados dos algoritmos construtivos consultados, e aos melhores resultados conhecidas para as instâncias com emprego de métodos de busca.

1 Introdução

O problema abordado neste trabalho é o posicionamento de formas irregulares bidimensionais, também conhecido como corte de estoque bidimensional ou aninhamento de polígonos. O cenário do problema é constituído de uma coleção de elementos, agrupados em um conjunto de itens e um conjunto de objetos. A tarefa em observação é a organização automatizada dos itens, posicionados sobre uma superfície supostamente plana de cada objeto disponível. E para este posicionamento, apenas uma representação plana de cada item é utilizada. O objetivo é que a organização feita proporcione um valor mínimo para a área total de regiões das superfícies dos objetos que não foram ocupadas por itens posicionados. Esta organização dos itens sobre os objetos será denominada leiaute. Por simplicidade, será considerada neste trabalho uma variação do problema que trata apenas de um objeto para receber todos os itens necessários, sendo que

este objeto é apresentado na forma retangular, com um dos lados limitado a um tamanho fixo, denominado largura, e o outro lado, denominado comprimento, possui tamanho suficiente para comportar qualquer distribuição dos itens. Este subproblema do posicionamento de formas irregulares foi classificado em [15] como problema de dimensão aberta, aqui especificamente em duas dimensões.

Para apresentar o método construtivo desenvolvido para fazer o leiaute, este artigo será estruturado como segue. A segunda seção traz uma definição do problema, contemplando a representação dos dados de entrada, o objetivo do processo e as restrições impostas. A terceira seção é dedicada a descrever o método construtivo proposto, indicando as técnicas encontradas na literatura que foram combinadas para desenvolvê-lo. Na quarta seção são apresentados os testes realizados e os resultados obtidos. Por fim, são feitas algumas considerações sobre o trabalho realizado e perspectivas de investigações consequentes.

2 Definição do problema

Seja uma coleção P de n_p polígonos envolventes de formas irregulares que representam uma perspectiva planar dos itens que devem ser posicionados. A codificação específica para cada polígono p_i é composta de três elementos, que são um ponto de referência r_i, expresso como um par de coordenadas do espaço euclidiano em $\mathbb{R}^2$; uma sequência V_i de n_{v_i} pontos, $V_i = \{v_{i,1}, ..., v_{i,n_{V_i}}\}$, sendo cada ponto $v_{i,k}$ definido no sistema de coordenadas cuja origem é o ponto r_i, que constituem os vértices do polígono ordenados em sentido anti-horário; e um conjunto A_i de n_{a_i} ângulos, $A_i = \{\theta_{i,1}, ..., \theta_{i,n_a}\}$, que podem ser usados para rotacionar o polígono em seu próprio eixo. Seja também um valor w que define a largura do objeto de forma retangular que receberá os itens posicionados. Este objeto possui comprimento inicialmente indefinido, e será considerado capaz de comportar qualquer leiaute válido.

Um posicionamento l_{i,j_i} para um polígono i é definido como um par de valores $(\theta_{l_{i,j_i}}, t_{i,j_i})$. O primeiro é o ângulo com que deve ser aplicada uma rotação no polígono em seu próprio eixo, ou seja, $\theta_{l_{i,j_i}} \in A_i$. O segundo valor é um ponto no sistema de coordenadas de posicionamento, para onde o polígono deve ser transladado. Um leiaute é, portanto, um conjunto formado por uma combinação de posicionamentos $L_j = \{l_{1,j_1}, ..., l_{n_p,j_{n_p}}\}$. O índice j aponta uma entre todas as diferentes combinações de posicionamentos possíveis. O objetivo neste problema é determinar uma combinação que leve a um valor mínimo do comprimento usado do objeto.

A altura h_i de um polígono p_i posicionado é a maior dentre as distâncias $h'_{i,k}$, que por sua vez representam as menores distâncias entre cada vértice $v_{i,k} \in V_i$ e a reta suporte para a base do objeto, após as operações de rotação e translação do posicionamento l_{i,j_i}. Desta forma, o comprimento do leiaute é $H(L_j) = h'_{i,m}$, tal que $h'_{i,m} \geq h'_{i,k}, \forall k \in \{1, .., n_{v_i}\}$. Resta diferenciar leiautes válidos de inválidos.

Dois critérios são usados para verificar a validade de um leiaute: a sobreposição de polígonos e o posicionamento de algum vértice de um polígono em uma região externa ao objeto que recebe as formas. A verificação do primeiro critério

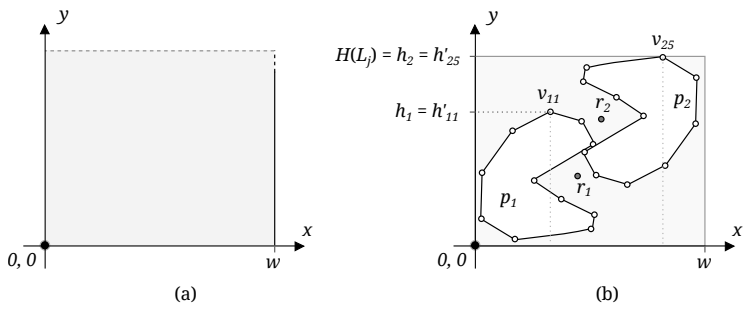

Figura 1. Representações do objeto receptor (a) e de posicionamentos (b).

é feita observando se para todos os vértices $v_{i,k} \in V_i$, onde $v_{i,k} = (x_{i,k}, y_{i,k})$, é verdade que $0 \leq x_{i,k} \leq w$ e $y_{i,k} \geq 0$. Porém, a verificação da ocorrência de sobreposição de polígonos e, principalmente, a identificação prévia de coordenadas que, se usadas para transladar um polígono para um leiaute, não levem à sobreposição deste polígono a um ou mais polígonos já posicionados, fazem partes de propostas de métodos para solucionar o problema de posicionamento de formas irregulares, disponíveis na literatura especializada, e serão comentadas na subseção 3.2.

3 Método proposto para construção de um leiaute

De acordo com o exposto em [11], determinar se uma combinação de posicionamentos leva a um leiaute de comprimento mínimo para uma instância é um problema que pertence à classe NP-completo, o que leva à busca por soluções plausíveis, não necessariamente ótimas, em abordagens que respondam em tempo suficientemente curto para elevar sua aplicabilidade prática. Dois grupos de abordagens são comumente avaliadas em problemas da classe NP-completo, que são as heurísticas construtivas e os métodos baseados em pesquisas locais, como afirmado em [9]. As heurísticas construtivas para o posicionamento de formas irregulares constituem-se de dois subprocessos. Um deles é a ordenação do conjunto de itens que devem ser posicionados, de acordo com algum critério. O outro é a regra de posicionamento, que determina quais serão os valores da posição e do ângulo de rotação aplicados a cada item. O método descrito a seguir é uma heurística construtiva inovadora por ser baseada na ordenação dos itens e na aplicação intercalada de dois subprocessos: uma regra gulosa para posicionar um certo subconjunto dos itens, e um método de melhoria da solução parcial com programação linear.

Para construir leiautes de qualidade satisfatória, ou seja, com uma taxa aceitável de não aproveitamento do objeto receptor, será utilizado o algoritmo guloso definido de acordo com a declaração exibida na figura 2. Os parâmetros deste algoritmo são:

1. P : conjunto de polígonos que devem ser posicionados;

```
01  Constrói_Solução_PFI (P, w, ordemPrimária, ordemSecundária,
02                         regra, opçãoAgrupamento,
03                         intervaloEntreCompactações)
04  Início
05    L := {};
06    contaAgrupamento := 0;
07    para i := {1,...,n} faça
08      item := Seleciona_Próximo_Item(ordemPrimária, ordemSecundária);
09      pos := Seleciona_Posicionamento_Válido(regra, item, L, w);
10      L := L + Aplica_Posicionamento(item, pos);
11      se (opçãoAgrupamento = porItens) ou
12         (opçãoAgrupamento = porFormas e
13          Quantidade_Itens_Restantes_Forma(item) = 0) então
14        contaAgrupamento := contaAgrupamento + 1;
15      fim_se
16      se contaAgrupamento > intervaloEntreCompactações então
17        contaAgrupamento := 0;
18        Compacta_Leiaute_Parcial(L, w);
19      fim_se
20    fim_para
21    se contaAgrupamento > 0 então
22      Compacta_Leiaute_Parcial(L, w);
23    fim_se
24    retorne L;
25  Fim.
```

Figura 2. Algoritmo para a construção de leiaute.

2. w : largura do objeto receptor;
3. *ordemPrimária* : critério usado para ordenar os polígonos de P;
4. *ordemSecundária* : critério usado para a ordenação, em casos de empate pelo critério de ordenação primária;
5. *regra* : identifica o critério usado para selecionar um posicionamento, dentre os possíveis;
6. *opçãoAgrupamento* : determina como os polígonos serão agrupados durante o posicionamento;
7. *intervaloEntreCompactações* : determina o tamanho do agrupamento que será formado antes que uma operação de compactação seja executada.

Há cinco sub-rotinas sendo usadas pelo algoritmo de construção de leiaute. Dessas, duas são relativamente mais simples que as demais. A primeira é denominada *Aplica_Posicionamento* e cuida de efetuar duas operações sobre o polígono selecionado, uma rotação com ângulo θ_{l_i}, e uma translação para o ponto t_{l_i}. A segunda, identificada por *Quantidade_Itens_Restantes_Forma*, apenas verifica quantos itens ainda restam para posicionar, que sejam representados por polígonos congruentes ao que representa o item que foi posicionado na iteração corrente. As outras sub-rotinas serão descritas mais detalhadamente nas subseções seguintes.

3.1 Heurísticas para seleção do próximo item

No algoritmo proposto, o leiaute é construído por uma sucessão de posicionamentos dos polígonos do conjunto P. Empregando as regras de posicionamento

que estão descritas na subseção 3.2, é importante observar que a variação da sequência dos itens posicionados interfere no comprimento final atingido. Com base nos trabalhos consultados, sobretudo [14] e [9], foram selecionados alguns critérios de ordenação que se valem da observação de algumas características dos polígonos. Na maior parte desses critérios, a ideia principal é posicionar os itens maiores primeiro, havendo variação na definição desta relação "maior que". Os critérios selecionados foram: comprimento decrescente (S_L), área decrescente (S_AR), concavidade decrescente (S_C) e retangularidade crescente (S_NR), de [14]; e perímetro decrescente (S_P), largura decrescente (S_W) e valor agregado decrescente (S_AG), de [9]. Esses critérios são classificados em [9] como "estáticos", pois independem do leiaute parcialmente construído. Foi também aproveitado um dos critérios "dinâmicos" proposto em [9], que consiste em selecionar o polígono que tenha a menor relação entre a contribuição para o comprimento do leiaute e a sua própria área (D_L).

Dos critérios estáticos considerados, há três que precisam ser definidos por não serem evidentes. A retangularidade é o inverso da diferença entre a área do retângulo envolvente do polígono e sua área interna, sendo que este retângulo tem a mesma largura e o mesmo comprimento do polígono, e sua base é paralela à base do objeto receptor. A concavidade é a diferença entre a área do invólucro convexo de um polígono e a sua área interna. Por fim, o valor agregado é obtido pela soma das posições de cada polígono nas ordenações feitas de acordo com os critérios estáticos descritos acima. Todos esses critérios, e os citados anteriormente, fornecem um item a ser posicionado, e a determinação do ponto e do ângulo de rotação que serão aplicados são tratados a seguir.

3.2 Regras de posicionamento consideradas

Foi obtida de [9] a técnica usada para determinar um conjunto finito de pontos viáveis para o posicionamento de um polígono qualquer no leiaute em construção, ou seja, sem incorrer na sobreposição a algum polígono já posicionado, além de garantir que todos os seus vértices estejam no interior do objeto receptor.

Naquele trabalho, o autor descreve operações que reduzem a quantidade de pontos aceitáveis a um número significativamente pequeno, tendo como base os conceitos de invólucro de posicionamento entre dois polígonos e de retângulo de posicionamento interior. Este último, denominado *inner-fit rectangle* (*IFR*) em [10], representa os limites aos quais o ponto de referência de um polígono está restrito para translação. Transladar um polígono por seu ponto de referência para algum ponto em uma região externa ao *IFR* faz com que algum vértice do polígono fique fora dos limites do objeto. Já o invólucro de posicionamento é também denominado polígono de obstrução em [13], e conhecido pelo termo *no-fit polygon* (*NFP*), que foi introduzido em [1] *apud* [6]. A figura 3 apresenta (a) um exemplo de *NFP* entre dois polígonos p_i e p_j, e (b) um exemplo de *IFR* para um polígono p_j. O polígono $NFP_{i,j}$ é tal que, se o polígono p_j for transladado para qualquer ponto de qualquer aresta de $NFP_{i,j}$, usando como referência o ponto r_j, então p_j estará tocando p_i, sem haver sobreposição entre eles. O fato de haver o toque entre arestas dos polígonos faz com que, nos pontos em que há

o toque, as distâncias entre os polígonos sejam zero, o que tende a ser favorável ao maior aproveitamento do objeto receptor.

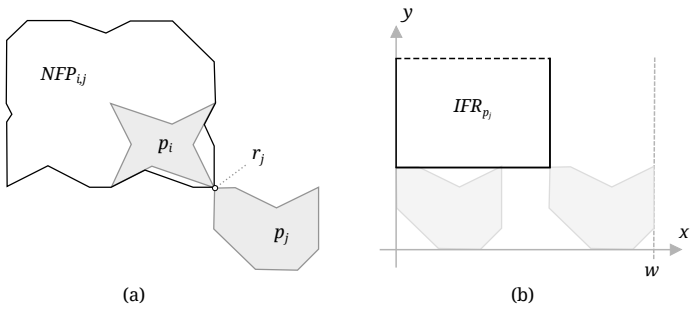

Figura 3. (a) Exemplo de *NFP* entre dois polígonos p_i e p_j; (b) exemplo de *IFR* para o polígono p_j.

Embora seja uma ferramenta poderosa para fornecer pontos de posicionamento, o cálculo do *NFP* ainda produz um conjunto infinito de possibilidades. A técnica exposta em [9] produz uma coleção finita de posições, sendo definido que cada ponto válido para posicionar um polígono p_j deverá atender a pelo menos uma das condições enumeradas a seguir, onde $NFP_{i,j}(x_i, y_i)$ representa o invólucro de posicionamento entre p_i e p_j transladado para o ponto de referência r_i, que está posicionado em (x_i, y_i). As condições são:

1. Ser um dos vértices do $NFP_{i,j}(x_i, y_i)$, para algum p_i já posicionado;
2. Ser um ponto de interseção entre uma aresta do $NFP_{i,j}(x_i, y_i)$ e uma aresta do $NFP_{k,j}(x_k, y_k)$, para quaisquer p_i e p_k já posicionados, $k \neq i$;
3. Ser um dos vértices do IFR_j;
4. Ser um ponto de interseção entre uma aresta do $NFP_{i,j}(x_i, y_i)$, para algum p_i já posicionado, e uma aresta do IFR_j.

Além disso, é acrescentado em [9] que qualquer ponto de posicionamento para p_j deve estar no exterior de $NFP_{i,j}(x_i, y_i)$, para todo p_i posicionado em (x_i, y_i), e no interior do IFR_j. Desta forma, tem-se uma coleção finita de pontos válidos, restando decidir qual desses será selecionado para o próximo item. No presente trabalho, foram utilizadas duas regras para fazer essa seleção: *bottom-left* e *middle-left*.

Encontrada nas abordagens encontradas em [7], [8] e [5], entre outras, a regra conhecida como *bottom-left* seleciona, dentre os pontos válidos, aquele que possui a menor componente y. Em caso de empate, prevalece o ponto que possui a menor componente x. Por sua vez, a regra *middle-left* seleciona o ponto que possui o menor valor absoluto de y. Uma observação importante é que, para esta regra, o *IFR* deve ser considerado como um retângulo que possui vértices de componentes y em $-\infty$ e $+\infty$. Assim, para esta regra, os vértices usados

na condição 3, apresentada anteriormente, deverão ser os pontos de interseção entre as arestas do *IFR* e a base do objeto receptor. O critério de desempate da regra *middle-left* é o mesmo da regra anterior. Ao final dos posicionamentos, todos os itens são transladados para um ponto $r_i^* = (x_i, y_i + y^*)$, onde $y^* = abs(min\{y_{i,k} \in V_i, \forall i \forall k\})$ Esta segunda regra foi proposta com o intuito de fornecer apenas uma base para comparações dos resultados da primeira. A seguir, é apresentado o procedimento de compactação utilizado.

3.3 Compactação de leiaute com programação linear

Em muitos casos, a escolha gulosa de uma posição feita de acordo com os critérios apresentados não corresponde à que leva ao melhor resultado possível. Visando melhorar os aproveitamentos dos leiautes produzidos, foi aplicado um método de compactação com programação linear obtido de [10], sendo este, por sua vez, uma adaptação do método proposto em [12]. Neste último trabalho se apresenta um modelo de compactação baseado em simulações de física, com velocidades e intervalos de tempo, buscando reduzir a energia potencial acumulada no modelo. Aqueles autores observam o elevado consumo de tempo desta abordagem e propõem uma melhoria neste aspecto, apresentando um modelo de compactação de leiautes baseado apenas em posicionamentos, sem simulações nem intervalos de tempo. Uma adaptação deste método é apresentada em [10], que modifica o objetivo da proposta original para minimizar o comprimento do leiaute. Nesses dois métodos, os modelos de programação linear incluem restrições que evitam a ocorrência de posicionamentos inválidos, que são sobreposições de itens ou desrespeito aos limites do objeto receptor.

Como definido na seção 2, o comprimento do leiaute é o valor da maior componente y dentre todos os vértices de todos os polígonos posicionados. A função objetivo de compactação pode ser definida como uma variável auxiliar z, que deve estar sujeita a restrições que garantem que $\forall v_{i,k} = (x'_{i,k}, y'_{i,k}), v_{i,k} \in V_i$ de um polígono $p_i(x_i, y_i)$, ou seja, transladado para o ponto (x_i, y_i), seja verdadeira a proposição $z \geq y'_{i,k}$. As coordenadas $(x'_{i,k}, y'_{i,k})$ representam os valores obtidos para os vértices $v_{i,k}$ convertidos para o sistema de coordenadas do objeto receptor. É preciso também garantir que os pontos de referência dos polígonos p_i estejam no interior do IFR_i. Sendo este retângulo definido por dois vértices, já que a altura será suficientemente grande para comportar qualquer posicionamento, então $IFR_i = \{(X^*_{i,0}, Y^*_{i,0}), (X^*_{i,1}, Y^*_{i,0})\}$, onde $X^*_{i,0}, X^*_{i,1}, Y^*_{i,0}$ são, respectivamente, o afastamento mínimo aceitável da lateral esquerda do objeto receptor para o ponto de referência r_i, o afastamento máximo aceitável da mesma lateral para o mesmo ponto, e o afastamento mínimo aceitável da base do objeto para o mesmo ponto de referência. Basta incluir restrições que assegurem que, sendo $r_i = (x_i, y_i)$, então as proposições $x_i \geq x_{i,0}$, $x_i \leq x_{i,1}$, $y_i \geq y_{i,0}$ devem ser verdadeiras. Deve-se observar que para a regra de posicionamento *middle-left*, não é necessário restringir y_i a um valor mínimo.

Assegurar que não haja sobreposições é, de fato, o ponto chave da elaboração deste modelo de compactação. Usar relações entre o ponto de referência e todas as arestas do *NFP* para cada par de polígonos traria para o modelo disjunções

de restrições, e consigo variáveis binárias, que provocariam uma demanda significativa de tempo para solucioná-lo. Visando simplificar o modelo, mantendo apenas variáveis contínuas, em [12] propõe-se o uso de uma heurística de localidade, que para cada par de itens p_i e p_j seleciona uma aresta do $NFP_{i,j}$ que seja mais "próxima" do segmento de reta $\overline{r_i r_j}$. Por sua vez, em [10] descreve-se um método de seleção de uma aresta do $NFP_{i,j}$ que tenha o ponto r_j sobre a aresta ou à sua direita[1], e selecionando, das arestas que satisfazem esta condição, aquela que tenha sua reta suporte mais distante de r_j. Uma restrição deve garantir que r_j permaneça sobre a aresta selecionada ou à direita desta. Nos casos em que esta aresta pertence a uma concavidade do $NFP_{i,j}$, a manutenção de r_j sobre ou à direita de todas as arestas que formam esta concavidade deve ser incluída no modelo como uma conjunção de restrições. Em [10], está incluído no modelo um conjunto de restrições que limitam a distância máxima entre as coordenadas do ponto de posicionamento atual de um item p_j e do ponto definido pela resolução do problema de programação linear a valores DX_j e DY_j. É estabelecido, em [9], um valor para este limite em 50% do comprimento do retângulo envolvente do item para DX, e 50% da largura do retângulo envolvente para DY.

A resolução do modelo de compactação fornece um par (x_i, y_i) para cada item p_i, que indicam as coordenadas do ponto para onde deve ser transladado p_i. Aplicando a transformação a cada item posicionado, obtém-se um leiaute de comprimento menor ou igual ao de antes da compactação. Em [10], define-se um processo iterativo que faz sucessivas compactações, enquanto houver redução do comprimento do leiaute.

Neste trabalho, foram empregadas duas maneiras de compor grupos de itens. Uma é por congruência das formas dos itens, que permite posicionar todos os itens de uma mesma forma, ou todos os itens de um subconjunto de formas, antes de aplicar uma nova compactação. A outra maneira é por ocorrência simples na sequência fornecida pelo método heurístico de seleção do próximo item. Esta segunda permite que sejam posicionados itens em uma determinada quantidade, não importando as formas destes, antes que seja aplicada nova compactação.

A figura 4 ilustra a aplicação da compactação intercalada com posicionamentos de subconjuntos de itens. A figura apresenta soluções parciais rotacionadas 90º para a direita, em relação à definição do sistema de coordenadas de posicionamentos no objeto receptor, ou seja, com o ponto de origem dos eixos de coordenadas posicionado no canto superior direito de cada solução parcial. Nesta figura, dividida em 12 partes, tem-se o posicionamento de cinco itens fornecidos pela heurística de seleção (i); em seguida, é efetuada uma compactação do leiaute (ii); esta sequência intercalada de posicionamentos e compactações continua até que todos os itens estejam posicionados (ix) e a última compactação tenha sido efetuada já sobre o leiaute completo (x). Para comparativo, são apresentados ainda dois leiautes produzidos com as mesmas configurações do algoritmo, porém sem a aplicação da compactação (xi), e outro aplicando uma única vez a

[1] Naquele trabalho, os autores utilizam a ordenação das arestas do *NFP* em sentido horário, fazendo com que o ponto r_j precise estar sobre uma dessas arestas ou à esquerda delas.

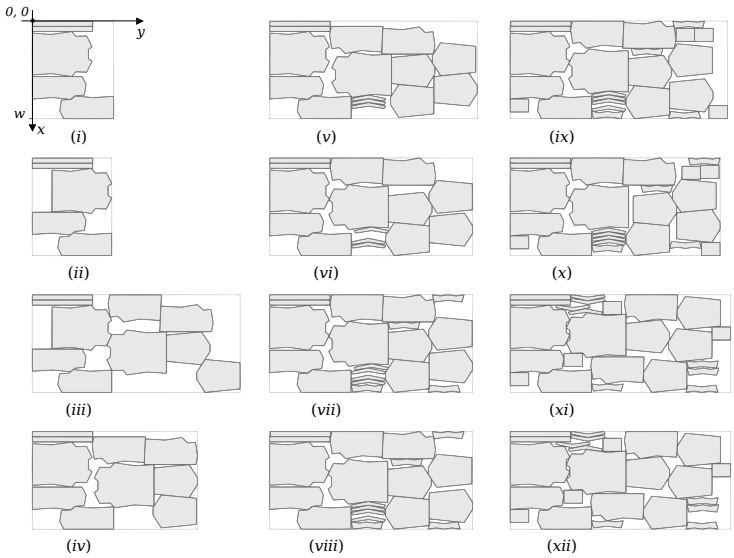

Figura 4. Produção de leiautes parciais com posicionamentos e compactações.

compactação ao final da construção do leiaute (*xii*). Nota-se, nestes dois últimos leiautes, que o algoritmo de compactação, por não alterar a posição relativa de cada par de itens, não consegue reorganizar os itens para reduzir o comprimento.

É possível observar neste exemplo que o diferencial deste algoritmo reside na capacidade de reduzir iterativamente a contribuição do posicionamento dos itens para o aumento do comprimento do leiaute. Isto é feito através da movimentação interna das soluções parciais, reajustando os itens posicionados. Esta capacidade, porém, é influenciada por características das instâncias e por variações das opções de execução do método. Os testes que foram executados a uma seleção de instâncias e combinando diferentes valores para os parâmetros do algoritmo proposto estão relatados na próxima seção.

4 Testes realizados e avaliação de resultados

Como ferramenta para avaliar o algoritmo proposto, foi feita uma implementação capaz de executá-lo para obter soluções para instâncias do problema abordado. Os subprogramas para a resolução dos modelos de programação linear fazem uso da biblioteca *lpsolve*, [4]. Os subprogramas necessários para os cálculos geométricos, por outro lado, foram implementados. Entre esses, o cálculo do NFP foi baseado no processo descrito em [2], que aplica o princípio das somas de Minkowski tendo cuidado peculiar com o consumo de tempo da execução. Diferentemente do processo apresentado no trabalho daqueles autores, o subprograma para o cálculo do NFP implementado não trata dos ciclos internos gerados pelas "somas" das arestas dos polígonos, ignorando-os. Consequente-

mente, os NFPs obtidos podem causar o desperdício de regiões viáveis para o posicionamento. No entanto, regiões inviáveis não são indevidamente ocupadas, preservando a aceitabilidade dos resultados. Esta limitação deste subprograma fez com que o algoritmo proposto não pudesse ser adequadamente avaliado para instâncias nas quais há ocorrência de pares de polígonos que levam a NFPs com furos.

Para realização de testes, algumas instâncias são comumente utilizadas, possibilitando proceder comparativos entre métodos de posicionamento de formas irregulares. Estas instâncias foram aproveitadas neste trabalho, tendo sido obtidas diretamente do *web site* do Grupo Europeu de Interesse em Problemas de Corte e Empacotamento (*EURO Special Interest Group on Cutting and Packing*), abreviado pelo acrônimo ESICUP. Das instâncias disponíveis, foram selecionadas 14 para usar como entrada para o algoritmo de posicionamento proposto. O critério para a seleção baseou-se na limitação do método para o cálculo do NFP usado na implementação do algoritmo, sendo rejeitadas as instâncias que contêm casos de NFPs com furos. As instâncias que foram selecionadas são identificadas por: ALBANO, BLAZ2, DAGLI, DIGHE1, DIGHE2, FU, MAO, MARQUES, SHAPES0, SHAPES1, SHAPES2, SHIRTS, SWIM e TROUSERS. Cada uma dessas instâncias inclui dados que representam um conjunto de diferentes polígonos, com a indicação da quantidade de cópias dos itens que devem ser posicionadas e dos ângulos admissíveis para rotação de cada polígono, além da largura do objeto receptor. O algoritmo foi executado para cada instância uma vez para cada diferente combinação de parâmetros aplicável à instância. Como os números de itens e de diferentes formas de itens em cada instância varia, o número de testes aplicados também varia. Os números de unidades de testes realizados serão apresentados adiante. As variações dos valores dos parâmetros do algoritmo foram estabelecidas como informado a seguir:

1. *ordemPrimária*: para ordenação primária (prim), foram usados os critérios estáticos S_AR, S_P, S_L, S_W, S_C, S_NR e S_AG, e foi também usado o critério dinâmico D_L;
2. *ordemSecundária*: para decidir situações de empate pelo critério de ordenação primária, foram usados os critérios S_AR, S_P, S_L, S_W;
3. *regra*: foram usadas as duas regras de posicionamento descritas na seção 3.2, sendo que para a regra *middle-left* foram testadas variações de pontos considerados para selecionar uma posição entre as válidas para cada item, conforme descrição a seguir:
   (a) *ponto de referência* (BR): esta é a maneira usual, também usada na regra *bottom-left*;
   (b) *ponto central* (BC): considera o ponto central do interior do retângulo envolvente do polígono;
   (c) *extremidade oposta* (M_L): compara os maiores valores alcançados pela componente y, no sistema de coordenadas do objeto receptor, dos vértices do polígono.
4. *opçãoAgrupamento*: as maneiras que foram aplicadas para organizar agrupamentos de itens foram aquelas descritas na seção 3.3, que agrupam por

unidades de itens (un) ou por unidades de conjuntos de itens que possuem formas congruentes (sh);

5. *intervaloEntreCompactações*: foram usados agrupamentos sem intervalos entre as compactações, ou seja, aplicando esta sub-rotina a cada posicionamento de item, e intervalos variando de um até o máximo permitido pela composição de cada instância para a opção de agrupamento em teste.

Os leiautes construídos são avaliados qualitativamente através da medida relativa entre o somatório das áreas dos itens posicionados e a área do retângulo de largura igual à largura do objeto receptor, e comprimento igual ao comprimento do leiaute. Esta medida representa a taxa de aproveitamento de um leiaute, $A(L)$, e é calculada de acordo com a equação (1), onde s_i é a área interna do polígono p_i, w é a largura do objeto receptor, e $H(L)$ é o comprimento do leiaute construído.

$$A(L) = 100 \times \frac{\sum_{i=1}^{n} s_i}{w \times H(L)} \tag{1}$$

Os testes foram executados em um computador com processador *Phenom* 9550 2.2 GHz, 4 GB de memória RAM, executando o sistema operacional *Linux*. Os modelos de programação linear foram solucionados com a biblioteca *lp_solve* na versão 5.5.2.0. Para fins de comparativo do desempenho da aplicação do método proposto, foram selecionados três métodos construtivos disponíveis na literatura e que foram definidos para solucionar o mesmo problema em questão neste artigo. Estes métodos são comentados a seguir.

Um dos métodos construtivos selecionados para comparação foi obtido de [9] e é aqui referenciado por BL-G. Este usa um algoritmo que combina uma coleção de doze diferentes heurísticas disponíveis para seleção do próximo item e uma regra de posicionamento *bottom-left*, valendo-se do cálculo de *NFP* para obter posições válidas. Outro método selecionado é denominado TOPOS, e foi proposto em [14]. Para propor este método, os autores lidaram com o problema do melhor ajuste entre os itens num leiaute. Naquele estudo, foram definidas 126 variantes possíveis para o algoritmo, combinando diferentes estratégias de posicionamento e diferentes fórmulas para avaliar as soluções possíveis a cada passo, além de duas abordagens, que são busca local ou ordenação inicial. Por fim, foi selecionado o método resultante de uma revisão feita em [3] sobre o método TOPOS. Nesta revisão, os autores sugeriram melhorias para alguns componentes do método original, como o aproveitamento de espaços livres entre os itens já posicionados e novas formulações para o critério de avaliação das posições válidas. Este método será aqui denominado TOPOS-R.

A tabela 1 apresenta um resumo dos resultados obtidos com os testes. Os dados estão separados por instâncias, que estão organizadas uma a cada linha da tabela. As colunas representam, da esquerda para a direita, o nome da instância, o número de unidades de testes para cada instância, a soma dos tempos de execução de todos os testes realizados para a instância (em segundos), a média do aproveitamento obtido com os testes, o melhor resultado de aproveitamento obtido, o melhor resultado encontrado na literatura que tenha

sido obtido com um método construtivo, e o melhor resultado encontrado em publicações para cada instância, incluindo aí os resultados de métodos de buscas. É feito um comparativo entre os resultados obtidos com o método proposto e os melhores resultados de métodos construtivos publicados. As taxas de aproveitamento maiores para cada instância estão destacadas com um asterisco à direita. Os trabalhos consultados para reunir as informações sobre outros métodos estão citados abaixo da tabela, com um número identificador que permite a relação do resultado apresentado com o trabalho citado.

Tabela 1. Melhores resultados dos testes e dados da literatura.

		Tempo (s)	resultados obtidos		melhores da literatura	
instância	unid.	Σ t	média ($\overline{A}$)	melhor (A)	construtivo	geral
albano	1260	1510.88	74.92	82.58 *	80.90 [0]	88.39 [4]
blaz2	1142	771.28	66.30	71.92 *	-	68.60 [9]
dagli	1686	2352.82	71.56	80.63 *	78.83 [0]	87.97 [3]
dighe1	1254	595.80	63.65	77.97 *	68.78 [0]	100.00 [5]
dighe2	943	239.36	64.92	77.11 *	71.84 [0]	100.00 [5]
fu	1455	504.31	71.52	83.82 *	79.17 [0]	92.03 [7]
mao	1186	2498.24	70.50	80.06 *	72.82 [0]	85.15 [7]
marques	1252	2218.19	76.55	84.73 *	80.59 [0]	90.01 [4]
shapes0	1632	3171.76	55.16	60.45	61.39 * [0]	68.44 [6]
shapes1	1632	4076.05	57.38	64.88	67.60 * [2]	73.84 [7]
shapes2	2036	3085.69	72.22	79.41 *	74.74 [1]	84.25 [6]
shirts	4286	42906.10	82.33	87.01 *	84.10 [2]	89.69 [3]
swim	1795	41621.69	62.76	69.36 *	66.82 [0]	75.60 [8]
trousers	3584	28230.37	80.02	87.72 *	86.00 [2]	90.60 [8]

(0) Bottom-Left [Gomes, 2005];

(1) TOPOS [Oliveira e Gomes e Ferreira, 2000];

(2) TOPOS revisado [Bennel e Song, 2008];

(3) Beam-Search [Bennel e Song, 2008];

(4) ResolveDual [Sato, 2011];

(5) SAHA [Gomes, 2005];

(6) ILSQN [Imamichi e Yagiura e Nagamochi, 2007];

(7) 2DNest [Egeblad e Nielsen e Odgaard, 2007];

(8) AutoNester-T [Nielsen e Odgaard, 2003];

(9) Tabu-Search [Blazewicz et al., 1993 apud Burke et al., 2005].

Das instâncias avaliadas, apenas para SHAPES0 e SHAPES1 não foram obtidos resultados melhores do que os dos métodos construtivos consultados na literatura. Para essas duas instâncias, o subprocesso de compactação não se mostrou eficaz, conduzindo a resultados bem semelhantes ao posicionamento sem este subprocesso. Todavia, para as outras instâncias avaliadas o algoritmo atingiu resultados com boas taxas de aproveitamento. Os parâmetros de configuração do algoritmo, bem como o tempo em segundos para obter os melhores resulta-

dos, além do número de compactações efetuadas nestes testes, são exibidos na tabela 2. É importante ressaltar que o elevado número de compactações (comp) efetuadas para algumas instâncias, por exemplo TROUSERS, é consequência da adoção da técnica de compactação limitada e iterativa, como descrita anteriormente.

Tabela 2. Detalhes dos testes que obtiveram os melhores resultados.

instância	regra	op	prim	sec	gr	tempo	comp
albano	BL	BR	D_L	S_P	5 un	1.36	34
blaz2	ML	M_L	D_L	S_AR	13 un	0.63	13
dagli	ML	BR	D_L	S_P	14 un	1.46	20
dighe1	BL	BR	S_AR	S_L	5 sh	0.37	15
dighe2	ML	BC	S_AG	S_W	1 sh	0.31	26
fu	BL	BR	D_L	S_AR	1 un	0.66	50
mao	ML	BC	S_P	S_AR	3 un	2.73	36
marques	BL	BR	D_L	S_W	10 un	1.59	16
shapes0	ML	BC	S_NR	S_AR	2 un	6.24	89
shapes1	ML	BR	S_AR	S_P	5 un	4.04	39
shapes2	BL	BR	S_L	S_W	11 un	1.46	19
shirts	BL	BR	S_C	S_L	3 sh	5.77	19
swim	BL	BR	S_P	S_L	10 un	22.29	23
trousers	BL	BR	S_P	S_AR	2 un	27.61	151

Os dados apresentados na tabela 2 descrevem as aplicações do algoritmo que obtiveram os melhores resultados. Porém, os parâmetros desses testes não mostram alguma combinação de valores que domine os resultados. Considerando que os experimentos realizados consistiram em uma busca por combinações de parâmetros que levassem aos melhores resultados, pode-se descrever uma estrutura de vizinhança para os experimentos como as combinações que possuem valores iguais para algum parâmetro. A tabela 3 exibe valores de parâmetros que levaram às vizinhanças com as melhores médias de aproveitamento.

De acordo com os testes efetuados, observa-se que, embora o uso da regra *middle-left* tenha levado aos melhores dos resultados obtidos para seis instâncias, somente para duas destas essa regra não foi menos estável que a regra *bottom-left*. Caso semelhante ocorre com a aplicação dos critérios de seleção do próximo item. O critério dinâmico utilizado, a menor contribuição relativa para o aumento do comprimento do leiaute, colaborou para atingir os melhores resultados em cinco instâncias, enquanto o critério estático de seleção por maior área colaborou para apenas dois dos melhores resultados. Porém, em termos de obtenção das melhores médias de resultados para cada instância, a vizinhança de combinações de parâmetros com este critério estático de seleção por maior área foi a que obteve as melhores médias com seis instâncias, enquanto o critério dinâmico

Tabela 3. Parâmetros com melhores médias de resultados.

	regra-op		prim		grupo	
instância	param	$\overline{A}$	param	$\overline{A}$	param	$\overline{A}$
albano	BL-BR	75.76	S_AG	77.67	4 un	75.96
blaz2	BL-BR	66.95	S_AR	67.16	13 un	67.15
dagli	ML-BC	72.28	D_L	76.48	4 un	72.65
dighe1	BL-BR	65.49	S_AR	68.01	1 un	65.73
dighe2	ML-BC	66.92	S_P	68.09	8 un	66.02
fu	BL-BR	73.07	D_L	74.63	9 un	72.72
mao	ML-M_L	72.11	S_AR	72.42	1 un	71.99
marques	BL-BR	79.39	S_AR	80.56	1 un	78.19
shapes0	BL-BR	56.70	D_L	57.17	5 un	55.78
shapes1	BL-BR	59.33	S_AR	58.99	7 un	58.13
shapes2	BL-BR	74.26	D_L	72.51	1 un	73.38
shirts	BL-BR	84.10	S_NR	83.02	4 un	83.27
swim	BL-BR	64.08	S_L	63.51	5 un	64.04
trousers	BL-BR	83.29	S_AR	82.87	1 un	82.07

citado atingiu melhor média de resultados com três instâncias. Constata-se que há uma grande variação entre as melhores combinações de parâmetros e as combinações mais estáveis. Isto também pode ser verificado para a configuração dos agrupamentos de itens a posicionar entre cada compactação. Embora 50% dos melhores resultados apresentem agrupamentos com dez ou mais itens, apenas para uma instância o uso de agrupamento com mais de dez itens foi o que levou à vizinhança de melhor média de resultados.

5 Considerações finais

Este trabalho apresentou um método construtivo de posicionamento de formas irregulares baseado na combinação de regras de posicionamento e compactação de soluções parciais com programação linear. Os resultados mostram que este método é competitivo no contexto de abordagens gulosas. Além disso, os subprocessos que compõem este método podem ser modificados para incorporar diferentes critérios de ordenação e diferentes regras de posicionamento, permitindo adaptá-lo a condições peculiares de algumas aplicações específicas, bem como experimentar novas abordagens construtivas.

Algumas investigações adicionais são sugeridas para aprimorar o desempenho do algoritmo proposto. Aplicar critérios de seleção baseados nas estratégias presentes no algoritmo TOPOS pode elevar a probabilidade de atingir resultados satisfatórios, melhorando o ajuste entre os itens posicionados. Com relação ao consumo de tempo de processamento, é válido estudar uma forma de reaproveitar

alguns cálculos de posições válidas após as execuções da rotina de compactação. Ainda seria recomendável avaliar técnicas de agrupamento de itens que considerem características individuais dos itens e das instâncias por completo.

Referências

1. M. Adamowicz and A. Albano. Nesting two-dimensional shapes in rectangular modules. *Computer-Aided Design*, 8(1):27–33, 1976.
2. J.A. Bennell and X. Song. A comprehensive and robust procedure for obtaining the nofit polygon using Minkowski sums. *Computers & Operations Research*, (June):1–37, 2008.
3. J.A. Bennell and X. Song. A beam search implementation for the irregular shape packing problem. *Journal of Heuristics*, 16:167–188, 2010.
4. M. Berkelaar, K. Eikland, and P. Notebaert. lpsolve v5. 5.2.0: Open source (mixed-integer) linear programming system, 2010.
5. 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*, 54(3):587–601, May 2006.
6. 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*, 179(1):27–49, May 2007.
7. B. Chazelle. The bottom-left bin-packing heuristic: An efficient implementation. *IEEE Transactions on Computers*, pages 697–707, 1983.
8. K.A. Dowsland, S. Vaid, and W.B. Dowsland. An algorithm for polygon placement using a bottom-left strategy. *European Journal of Operational Research*, 141(2):371–381, 2002.
9. A. Gomes. *Abordagens Heurísticas ao Posicionamento de Formas Irregulares*. PhD thesis, Universidade do Porto, 2005.
10. A. Gomes and J. Oliveira. Solving Irregular Strip Packing problems by hybridising simulated annealing and linear programming. *European Journal of Operational Research*, 171(3):811–829, June 2006.
11. E. Hopper. *Two-dimensional packing utilising evolutionary algorithms and other meta-heuristic methods*. PhD thesis, University of Wales, Cardiff, 2000.
12. Z. Li and V. Milenkovic. Compaction and separation algorithms for non-convex polygons and their applications. *European Journal of Operational Research*, pages 539–561, 1995.
13. T.C. Martins. *Estudo do Recozimento Simulado e do Polígono de Obstrução Aplicados ao Problema de Empacotamento Rotacional de Polígonos Irregulares Não-Convexos em Recipientes Fechados*. PhD thesis, Escola Politécnica da Universidade de São Paulo, 2007.
14. J.F. Oliveira, A.M. Gomes, and J.S. Ferreira. TOPOS-A new constructive algorithm for nesting problems. *OR Spectrum*, 22(2):263–284, 2000.
15. G. Wascher, H. Hausner, and H. Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109–1130, December 2007.