André Kubagawa Sato

Proposta de Algoritmo para a Determinação da Região Livre de Colisão e sua Aplicação na Solução de Leiautes Bidimensionais Irregulares com Recozimento Simulado

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Engenharia.

São Paulo
2011

André Kubagawa Sato

Proposta de Algoritmo para a Determinação da Região Livre de Colisão e sua Aplicação na Solução de Leiautes Bidimensionais Irregulares com Recozimento Simulado

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Engenharia.

Área de concentração:
Engenharia Mecatrônica

Orientador:
Prof. Dr. Marcos de Sales Guerra Tsuzuki

São Paulo
2011

FICHA CATALOGRÁFICA

Sato, André Kubagawa
Proposta de algoritmo para a determinação da região livre de colisão e sua aplicação na solução de leiautes bidimensionais irregulares com recozimento simulado / A.K. Sato. -- São Paulo, 2011.
89 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos.

1. Otimização combinatória 2. Empacotamento e cobertura 3. Heurística I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos II. t.

A minha família, amigos e professores, que me deram grande apoio e incentivo durante a jornada acadêmica

Agradecimentos

Agradeço Prof. Dr. Marcos de Sales Guerra Tsuzuki pelas orientações durante a realização deste trabalho, por ter me guiado pelos caminhos da pesquisa, por seus ensinamentos e oportunidades oferecidas.

Agradeço o Prof. Dr. Thiago de Castro Martins pelos comentários sobre o texto e o grande auxílio no desenvolvimento da pesquisa.

Agradeço à minha família e amigos que me ajudaram na execução deste trabalho.

Resumo

O problema de empacotamento consiste em arranjar um conjunto de itens em um contêiner, a fim de maximizar sua utilização. Este campo de estudos tem impacto em diversas indústrias, incluindo as indústrias têxtil, moveleira e naval. Neste trabalho, dois problemas de empacotamento de itens irregulares são estudados. O primeiro, chamado primal, é o caso em que os itens possuem rotação livre e o contêiner de dimensões fixas pode ser representado por um polígono qualquer, podendo ser não convexo. O segundo problema, denominado dual, consiste em posicionar os itens, que possuem apenas algumas orientações possíveis, em um contêiner retangular em que uma das dimensões é considerada infinita. Assim, o objetivo é obter o menor contêiner, variando a dimensão não fixa, no qual todos os itens podem ser posicionados sem sobreposição. Em ambos problemas, a solução é representada por uma lista ordenada de itens e uma regra de posicionamento é aplicada para se obter o leiaute. Neste caso, sobreposições não são permitidas. Para se garantir leiautes factíveis (sem sobreposição), é adotado o conceito de região livre de colisão. A região livre de colisão representa todas as translações possíveis para inserir um novo item em um contêiner com itens já posicionados. A região livre de colisão é obtida através de operações Booleanas envolvendo polígonos de obstrução e de posicionamento interno. Devido às propriedades dos conceitos envolvidos, o cálculo da região livre de colisão deve ser feito utilizando operações Booleanas não regularizadas. Um novo algoritmo de operação Booleana não regularizada de união e subtração é desenvolvido a partir da implementação de um algoritmo de operações Booleanas regularizadas. Um algoritmo de recozimento simulado é utilizado para controlar a posição, o ângulo (ou orientação) e a sequência dos itens. Cada item só pode ser posicionado no vértice da região livre de colisão. Com a finalidade de melhorar o desempenho computacional do algoritmo, um método de paralelização do cálculo da região livre de colisão é proposto. Para comparação, são adotados dois algoritmos seriais. Através dos resultados, é possível afirmar que o algoritmo primal foi capaz de resolver problemas do tipo quebra–cabeça, incluindo contêineres convexos e com furos. O algoritmo apresentou melhora significativa no desempenho quando comparado com trabalhos anteriores. Para o caso dual foi proposto um algoritmo de dois níveis, em que o externo controla o comprimento do contêiner e o interno é semelhante ao primal. Este algoritmo foi testado com problemas existentes na literatura e apresentou soluções competitivas, obtendo alguns leiautes mais compactos. A paralelização apresentou ganho de desempenho apenas nos problemas com grande número de itens. Foi constatado que o custo computacional de operações Booleanas não regularizadas é fortemente dependente do número de vértices e intersecções dos polígonos de entrada da operação.

Abstract

The irregular shape packing problem is an optimization problem that consists of arranging items on a container in order to maximize the utility rate of the sheet stock. This work investigates two problems. In the first problem, the single bin packing, the items can rotate freely and the container with fixed dimension can be any polygon, convex or non-convex. The second problem, the open dimension problem, consists of arranging items that have few admissable orientations in a container with fixed width and variable length. The objective is to find a feasible layout of the set of items that minimizes the length of the container. The solution is always represented as an ordered list of itens to be packed and a placement heuristic is applied in order to generate a layout. To ensure feasible layouts, the concept of collision free region is adopted. It represents all the positions that a new item can be placed inside the container, without colliding with already placed items. The collision free region is obtained through non manifold Boolean operations applied to no-fit polygon and the inner-fit polygon. The simulated annealing algorithm controls the position, rotation and placement order of the items. Each item is is exclusively placed on collision free region's vertex. To improve the computational cost performance of the algorithm, a parallelization method to determine the collision free region is proposed. The speed of this algorithm is compared with two different serial methods of determing the collision free region. From the results, it can be observed that the solutions for the single bin packing problem are very competitive with previous works and can achieve optimal solution for puzzles with irregular shaped containers and containers with holes. The algorithm for the open dimension has two hierarchical levels: a core level with a simulated annealing algorithm, and the external level controlling the container length. This algorithm was tested with literature problems and obtained very competitive results, some which are more compact. The results showed that the parallelized version is better than the sequential approach only for datasets with very large number of items. The computational cost of the non manifold Boolean operation algorithm is strongly dependent on the number of vertices and intersections of the original polygons.

Conteúdo

Lista de Figuras

Lista de Tabelas

1 Introdução

Problemas de corte e empacotamento são problemas clássicos de busca pelo leiaute mais eficiente para um conjunto de itens de entrada e um contêiner com o objetivo de se minimizar o desperdício de material. As aplicações são numerosas, como por exemplo na indústria têxtil, naval, moveleira. Wäscher et al. [1] propôs uma tipologia que classifica os problemas de acordo com sua dimensão, objetivo, variedade de itens, assim como o número e a natureza do contêiner onde os itens serão inseridos. Neste trabalho, são estudados dois tipos de problemas. De acordo com a classificação mencionada, os problemas estudados são: problema bidimensional de itens irregulares com o objetivo de posicionar todos os itens no interior de um contêiner (two-dimensional irregular single bin) e problema bidimensional de itens irregulares em que o contêiner possui uma dimensão variável (two-dimensional irregular open dimension problem).

Apesar de existirem diversas propostas na literatura, é possível observar duas estratégias predominantes para representar e buscar o espaço de soluções. A primeira representa a solução como uma lista ordenada de itens e aplica uma regra de posicionamento para construir o leiaute [2–4]. A segunda estratégia representa a solução como um leiaute físico no contêiner e move os itens neste leiaute [5–7]. Esta estratégia permite a sobreposição de itens, impondo uma penalização na função objetivo, chamada penalização externa [8]. Martins e Tsuzuki [9] demonstraram que o uso de penalização externa pode resultar em uma solução ótima que possui leiaute com sobreposição, tornando a solução inválida.

O problema é NP-completo [10] e como consequência as soluções propostas usualmente se utilizam de heurísticas. Grande parte dos trabalhos publicados se utilizam de heurísticas probabilísticas [5, 9, 11, 12]. Outra abordagem de destaque é a utilizada em [6, 12], que utilizam uma combinação de heurísticas probabilísticas e determinísticas. Também foram propostos algoritmos baseados em programação linear [13, 14] e heurísticas "greedy" [15], que possuem melhor desempenho, no entanto são muito dependentes da configuração inicial.

Mais recentemente na área de empacotamento, foram divulgados trabalhos

que apresentaram uma grande melhoria nos resultados, obtendo leiautes com maior aproveitamento. Dentre estes trabalhos, Gomes e Oliveira [6] utilizam o algoritmo de recozimento simulado para guiar a busca no espaço de solução e utiliza algoritmos de compactação e separação baseados em programação linear. Bennell e Song [16] elaboraram uma versão aprimorada do algoritmo construtivo TOPOS, proposto por Oliveira et al. [17]. O algoritmo TOPOS procura por regiões fechadas definidas pelos itens já posicionados. Egeblad et al. [7] e Imamichi et al. [18] propuseram algoritmos que consideram contêineres de dimensão fixa e se utilizam de algoritmos de separação para se obter leiautes sem sobreposição. Se uma solução é encontrada, a dimensão variável do contêiner é reduzida. Para realizar uma separação, Egeblad et al. [7] adotaram uma função que indica o nível de sobreposição do item em função de sua posição, considerando apenas movimentos horizontais ou verticais. Imamichi et al. [18] desenvolveram algoritmos de compactação e separação através da programação não linear.

O recozimento simulado [19], que é um algoritmo de otimização, foi proposto para a área de otimização combinatória, em que a função objetivo é definida em um domínio discreto. O algoritmo foi modificado para ser aplicado na otimização de funções definidas em um domínio contínuo por Corana et al. [20] utilizando passos distintos de acordo com o intervalo de temperatura. Este trabalho utiliza o algoritmo de recozimento simulado proposto por Martins e Tsuzuki [21].

1.1 Motivação do estudo

Os problemas de corte e empacotamento têm grande impacto ecológico e econômico em diversas áreas. Isto porque o objetivo é sempre obter um melhor aproveitamento dos materiais envolvidos. Os primeiros problemas estudados neste campo foram os de cortes retangulares, devido à sua menor complexidade. Os problemas envolvendo itens irregulares apresentam um desafio maior. Nestes problemas, os especialistas humanos são capazes de obter leiautes compactos tão bons quanto os existentes na literatura. No entanto, o tempo para uma pessoa conseguir obter este leiaute pode variar muito [22]. Soluções computacionais podem obter tempos mais uniformes e se incorporar em um processo de produção que não necessite de intervenção humana.

A maior parte dos trabalhos publicados que tratam de itens irregulares são translacionais, admitindo algumas orientações para a peça. Isto pode ser explicado devido à demanda da indústria de tecidos, que permite apenas rotações de 180°, por causa das características do tecido. Assim, existe uma carência de

trabalhos de problemas de posicionamento irregular em contêineres de dimensões fixas e que permitem translações e rotações dos itens. Além disso, existem inúmeros problemas de corte de material nos quais o tratamento pelo posicionamento rotacional simplesmente não se aplica.

Um exemplo clássico de problema de corte de material é o caso da indústria têxtil, que raramente admite que um item possa ser rotacionado livremente, visto que este deve possuir um alinhamento pré–determinado com os padrões do tecido. Além disso, o comprimento da peça bruta de tecido é tão grande que este problema é melhor abordado com técnicas que tratam o contêiner com comprimento infinito [22].

Em outros casos, como o corte de metal na indústria mecânica, as peças a serem produzidas não possuem quaisquer restrições de rotação, e as peças brutas possuem dimensões limitadas. No entanto, ainda assim estes problemas de corte não são tratados como problemas de posicionamento rotacional, pois o custo de um processo de corte irregular é próximo do custo do material desperdiçado, sendo vantajoso considerá–los como problemas mais restritos, ou seja, como problemas de posicionamento retangular ou de corte de material por guilhotina [22].

Ainda assim, um estudo mais extensivo encontra diversos ramos da indústria nos quais a otimização de posicionamentos rotacionais com contêineres de dimensões fixas é de suma importância.

Considere o problema da indústria de couro. Nesta indústria, itens são recortados de peças brutas de couro. Verifica–se que neste caso o valor do custo do material desperdiçado supera facilmente o custo de um processo de corte irregular. Ainda mais grave, até a data presente, retalhos de corte de couro são classificados no Brasil como resíduos de classe C I (resíduos sólidos perigosos) [23], de modo que o seu descarte exige um processo específico de custo significativo.

A demanda da indústria por técnicas para o posicionamento rotacional em contêineres de dimensões fixas motiva este trabalho, que trata do problema de posicionamento em seu modo mais irrestrito, em que os itens são heterogêneos irregulares (convexos ou não–convexos), e seu posicionamento em contêineres irregulares (também convexos ou não–convexos) pode ser translacional ou rotacional.

1.2 Estrutura do trabalho

Este trabalho está dividido em quatro partes principais: definição do problema de empacotamento, apresentação dos conceitos de região livre de colisão e polígono

de obstrução, algoritmos propostos e análise dos resultados.

Após a introdução, o problema é definido no capítulo 2. São apresentados as características de complexidade (seção 2.1), função objetivo e dualidade (seção 2.2) e geometria (seção 2.5). Nas seções 2.3 e 2.4 são definidos os dois problemas de empacotamento estudados.

Os conceitos utilizados para garantir o posicionamento sem colisão em um contêiner são apresentados no capítulo 3. Os conceitos apresentados são: polígono de obstrução (seção 3.1), polígono de posicionamento interno (seção 3.2) e região livre de colisão (seção 3.3). Por fim a seção 3.4 apresenta dois métodos para se determinar a região livre de colisão, e um dos métodos é paralelizado. A descrição do algoritmo de operações Booleanas não regularizadas utilizadas no cálculo da região livre de colisão está contida no capítulo 4.

O capítulo 5 apresenta a decomposição do problema de posicionamento (seção 5.1) e os algoritmos propostos para resolver o problema primal (seção 5.4) e o dual (seção 5.5).

Os resultados são apresentados no capítulo 6. A seção 6.1 apresenta os resultados dos problemas primais, a seção 6.2 apresenta os resultados dos problemas duais da literatura e a seção 6.3 apresenta os tempos obtidos utilizando os algoritmos seriais e paralelos. Por fim a conclusão constitui o capítulo 7.

2 Descrição do problema a ser estudado

Neste capítulo serão apresentadas as características do problema de posicionamento translacional e rotacional em contêineres com todas as dimensões fixas ou com uma das dimensões variável.

Na seção 2.1 é discutida a questão da complexidade dos problemas de empacotamento. A seção 2.2 apresenta dois problemas de posicionamento estudados nesta pesquisa: o primal e o dual. São apontadas as dificuldades intrínsecas do problema de posicionamento em contêineres de dimensões fixas quando comparado à sua variante dual. As seções 2.3 e 2.4 definem os dois problemas, inclusive sua representação matemática. Na última seção deste capítulo, 2.5, é discutida a questão da geometria e de criação da leiautes factíveis.

2.1 Complexidade

É demonstrado que mesmo versões restritas do problema (por exemplo, que lidam apenas com retângulos) são NP-Difíceis, o que significa que não se acredita que seja possível resolver algoritmicamente o problema para um número prático de itens [10]. Por este fato diversos autores voltaram–se para abordagens baseadas em heurísticas probabilísticas, que não garantem a determinação da solução ótima, mas podem, em um tempo razoável, encontrar uma solução aceitável. Heurísticas probabilísticas de otimização seguem o seguinte padrão: enquanto um critério de parada não for satisfeito, em cada etapa avalia-se a função a ser otimizada em um conjunto de pontos e aplicam-se algumas regras para determinar o conjunto de pontos a ser avaliado na próxima etapa.

2.2 Características da função objetivo e dualidade

Adotando a mesma abordagem que [24], pode-se considerar que o posicionamento em contêineres de dimensões fixas possui um problema correlato dual que consiste em encontrar o menor contêiner capaz de conter todos os itens sem que haja sobreposição. No levantamento feito por Wäscher et al. [1], o problema dual encontra uma cobertura significativamente mais ampla na literatura.

Quando comparado à sua variante dual, o problema de posicionamento em contêineres de dimensões fixas possui uma característica que dificulta o seu tratamento, o fato de que ele é um problema de otimização de variáveis *contínuas* (translações e rotações dos itens) com uma função objetivo que assume valores *discretos* (área total não–ocupada pelos itens).

A Fig. 2.1 mostra um problema de otimização dual de uma só variável (a rotação θ do item mais a esquerda), no qual o objetivo é minimizar a largura total do contêiner $f(\theta)$. Como se pode ver, a largura mínima do contêiner varia de modo *contínuo* com a variável de otimização θ.

No caso do problema primal mostrado na Fig. 2.2, a função objetivo a ser minimizada é o espaço não ocupado no interior do contêiner. Como este só pode variar em função da área de itens sendo adicionados ou retirados do contêiner, a função objetivo assume somente uma quantidade finita de valores, tornando–se assim descontínua.

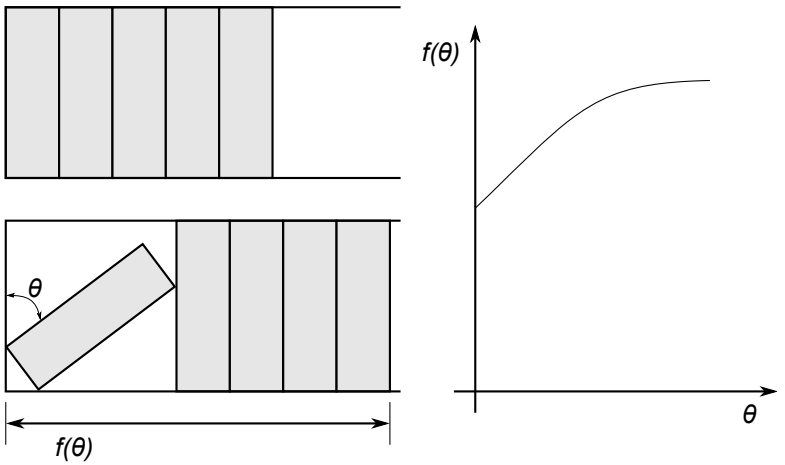


Figura 2.1: A variação da função objetivo no problema dual. Fonte: Martins [24].

Esta característica dificulta a avaliação da sensibilidade da função objetivo com relação às variáveis do problema.

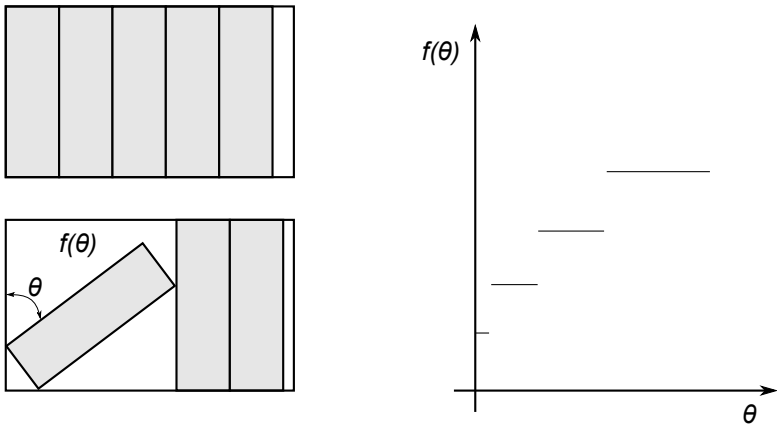


Figura 2.2: A variação da função objetivo no problema primal. Fonte: Martins [24].

2.3 Definição do problema primal

Um dos problemas estudados neste trabalho é o posicionamento rotacional de múltiplos itens em um único contêiner de dimensões fixas (problema primal). Ele pode ser definido como o problema de determinar um subconjunto de itens e suas transformações (translações e rotações) que os posicione dentro de um contêiner (polígono convexo ou não-convexo) sem sobreposição, de modo a maximizar a área ocupada (vide Fig. 2.3).

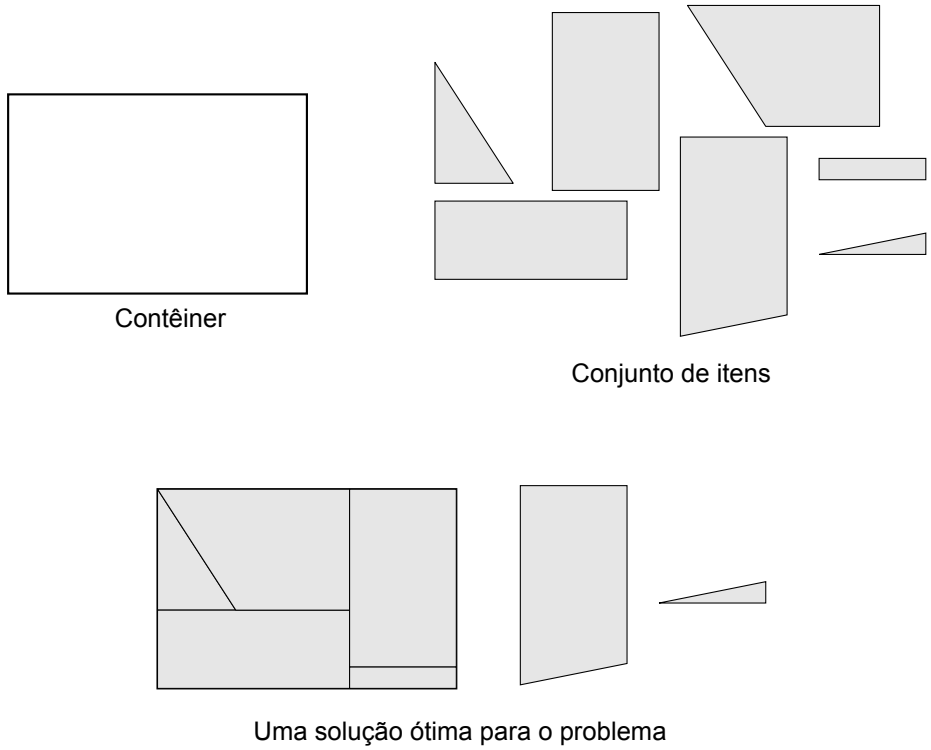


Figura 2.3: Um problema de empacotamento primal e sua solução ótima.

Definição 2.1 *Existe sobreposição entre itens quando a intersecção de seus interiores é não-vazia.*

É fornecida uma lista de polígonos $\mathcal{P} = \{P_1, \cdots, P_n\}$, que representam os itens, e um contêiner $\mathcal{C}$. Os polígonos de $\mathcal{P}$ e $\mathcal{C}$ podem ser não convexos.

Um polígono $P_i \in \mathcal{P}$ rotacionado de o é representado por $P_i(o)$, que pode ser escrito como P_i por simplificação quando a orientação não for especificada ou por já estar claro no contexto. Por conveniência, o polígono $P_i(o)$ com $(i = 1, \cdots, n)$ e o $\mathcal{C}$ representam um conjunto de pontos que inclui as suas regiões internas e contornos. Para um polígono S, seja $\imath(S)$ o interior de S e $A(S)$ sua área. As translações dos polígonos são representadas por meio de somas de Minkowski como segue.

Seja $\mathbf{x} = (x_{i1}, x_{i2})$ com $(i = 1, \cdots, n)$, um vetor translação para P_i. Então o polígono obtido pela translação do polígono P_i por $\mathbf{x}_i$ é $P_i \oplus \mathbf{x}_i = \{\mathbf{p} + \mathbf{x}_i \,|\, \mathbf{p} \in P_i\}$. A lista $\mathcal{E}$ possui os índices dos polígonos que foram posicionados. O problema primal do empacotamento irregular pode ser descrito formalmente como segue:

$$
\begin{aligned}
\text{minimizar} \quad & A(\mathcal{C}) - \sum_{i \in \mathcal{E}} A(P_i) \\
\text{restrições} \quad & \imath(P_i(o_i) \oplus \mathbf{x}_i) \cap (P_j(o_j) \oplus \mathbf{x}_j) = \emptyset \quad i, j \in \mathcal{E}, i \neq j \\
& (P_i(o_i) \oplus \mathbf{x}_i) \subseteq \mathcal{C} \quad i \in \mathcal{E} \\
& 0 \leq o_i < 2\pi \quad i \in \mathcal{E} \\
& \mathbf{x}_i \in \Re^2 \quad i \in \mathcal{E}
\end{aligned}
\tag{2.1}
$$

A solução do problema (2.1) é um vetor de translação $\mathbf{x} = (\mathbf{x}_1, \cdots, \mathbf{x}_n)$, um vetor de ângulos $\mathbf{o} = (o_1, \cdots, o_n)$ e uma lista de itens posicionados $\mathcal{E}$. Note que a solução $(\mathbf{x}, \mathbf{o}, \mathcal{E})$ determina o leiaute dos polígonos.

2.4 Definição do problema dual

O outro problema estudado é o posicionamento translacional de múltiplos itens em um único contêiner retangular que possui uma de suas dimensões variáveis. Ele pode ser definido como o problema de, dada uma das dimensões de um contêiner retangular, um conjunto de n itens (polígonos convexos ou não-convexos) e as orientações (ângulos pelos quais cada item pode ser rotacionado), determinar as translações e orientações que, ao serem aplicadas aos respectivos itens, os posicione dentro do contêiner sem sobreposições, e de modo a minimizar a dimensão variável (vide Fig. 2.4).

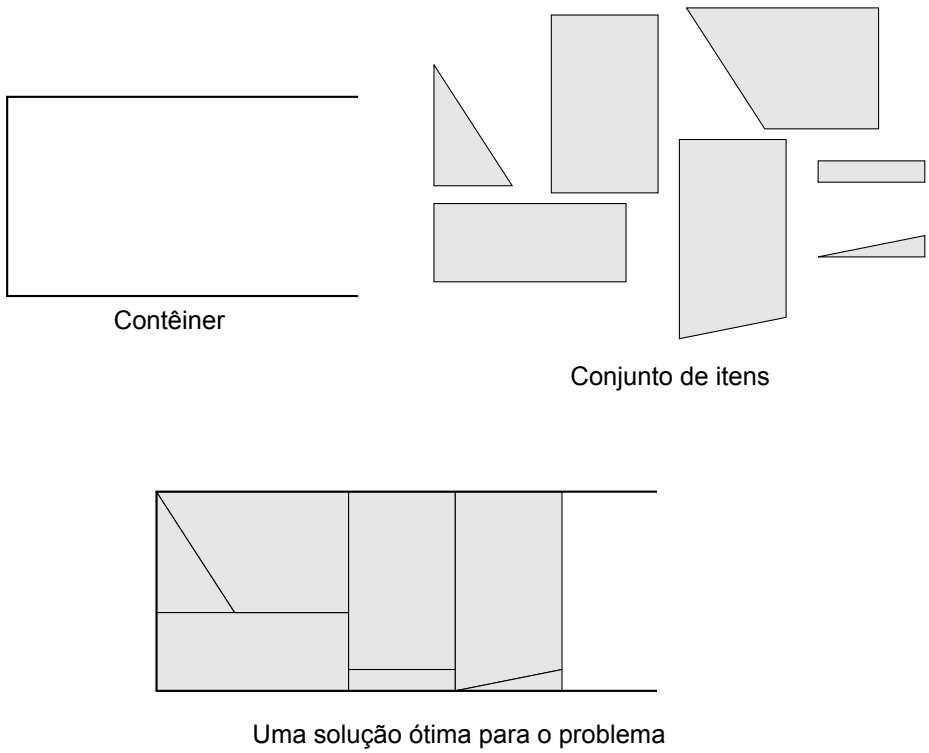


Figura 2.4: Um problema de empacotamento dual e sua solução ótima.

Para o problema dual é fornecida uma lista de polígonos $\mathcal{P} = \{P_1, \cdots, P_n\}$, uma lista de orientações de polígonos $\mathcal{O} = O_1 \times \cdots \times O_n$, onde O_i $(1 \leq i \leq n)$ representa um conjunto de orientações pelo qual o polígono P_i pode ser rotacionado, e um contêiner $\mathcal{C} = \mathcal{C}(W, L)$ de largura constante $W > 0$ e comprimento váriável $L > 0$. Os polígonos de $\mathcal{P}$ podem ser não convexos. Desta forma, o problema dual do empacotamento irregular pode ser descrito formalmente como segue:

$$
\begin{array}{ll}
\text{minimizar} & L \\
\text{restrições} & \imath(P_i(o_i) \oplus \mathbf{x}_i) \cap (P_j(o_j) \oplus \mathbf{x}_j) = \emptyset \quad 1 \leq i < j \leq n \\
& (P_i(o_i) \oplus \mathbf{x}_i) \subseteq \mathcal{C}(W, L) \quad 1 \leq i \leq n \\
& L \in \Re_+ \\
& o_i \in O_i \quad 1 \leq i \leq n \\
& \mathbf{x}_i \in \Re^2 \quad 1 \leq i \leq n
\end{array}
\tag{2.2}
$$

A solução do problema (2.2) é um vetor de translação $\mathbf{x} = (\mathbf{x}_1, \cdots, \mathbf{x}_n)$ e um vetor de orientações $\mathbf{o} = (o_1, \cdots, o_n)$. Note que a solução $(\mathbf{x}, \mathbf{o})$ determina unicamente o leiaute dos polígonos. O mínimo comprimento L do contêiner C é formalmente definido por uma função

$$\begin{aligned}\mu(\mathbf{x},\mathbf{o}) &= \max\{x_1 \,|\, (x_1,x_2) \in P_i(o_i) \oplus \mathbf{x}_i, P_i \in \mathcal{P}\} - \\ &\quad \min\{x_1 \,|\, (x_1,x_2) \in P_i(o_i) \oplus \mathbf{x}_i, P_i \in \mathcal{P}\} \end{aligned} \tag{2.3}$$

2.5 Geometria

Uma das características mais importantes dos problemas de empacotamento com itens irregulares é a necessidade de se empregar ferramentas geométricas avançadas. Mais especificamente, é necessária a utilização de algum dispositivo que possibilite detectar se, dada a posição de dois itens, eles se interseccionam, sobrepõem ou estão separados. Existem basicamente três ferramentas empregadas na literatura: método de rasterização, trigonometria direta e polígono de obstrução.

O método de rasterização divide o espaço em áreas discretas, diminuindo a informação geométrica. Mais de uma variação deste método foi desenvolvida [2, 25] e em todas a determinação da sobreposição é realizada checando apenas as células da grade. Este método possui um alto desempenho, no entanto necessita de muita memória e possui baixa precisão na representação de arestas não ortogonais.

Se considerados os itens como polígonos, é possível checar a colisão utilizando trigonometria direta. Deste modo, é necessário verificar o cruzamento das arestas dos polígonos. Neste caso, a precisão é muito maior que a do método anterior. No entanto, a complexidade do algoritmo é exponencial em relação ao número de arestas [26].

A ferramenta geométrica mais utilizada recentemente é o polígono de obstrução. A razão sua popularidade se deve ao fato de possuir precisão igual ao do método da trigonometria direta, sendo mais eficiente computacionalmente [26]. Neste trabalho, o conceito de polígono de obstrução é utilizado para definir todas as posições possíveis para a inserção de um novo item no contêiner.

3 Polígonos de obstrução e região livre de colisão

A construção de leiautes sem sobreposição é uma das dificuldades que se apresenta ao resolver o problema de empacotamento. Alguns trabalhos tratam o problema menos restrito, com sobreposição, aplicando uma penalização para obter uma solução sem colisão. No entanto, a solução ainda pode convergir para leiautes com sobreposição. Para garantir leiautes factíveis, o conceito de polígono de obstrução é utilizado [24]. Este conceito é o mais utilizado na literatura para evitar colisão entre dois itens. Em um leiaute com diversos itens já posicionados, para o posicionamento de um novo item é utilizada a região livre de colisão [24, 27]. Como existem diferentes maneiras de calcular a região livre de colisão, dois métodos foram propostos, sendo um paralelizado.

Na seção 3.1 são apresentados: o polígono de obstrução, suas propriedades e o método adotado para realizar seu cálculo. A seção 3.2 introduz o conceito do polígono de posicionamento interno, derivado do polígono de obstrução. É definida a região livre de colisão, na seção 3.3, e como obtê-la a partir do polígono de obstrução e do polígono de posicionamento interno. Por fim, o estudo dos métodos para calcular a região livre de colisão e sua paralelização é vista na secção 3.4.

3.1 Polígono de obstrução

O polígono de obstrução (No-Fit Polygon) é utilizado para assegurar leiautes factíveis em problemas de empacotamento de peças irregulares. Art [28] foi o primeiro a empregar este conceito que desde então vem sendo utilizado por diversos autores.

O polígono de obstrução representa translações aplicadas a itens, que são matematicamente simbolizadas por um conjunto de vetores. Para melhor compreensão das propriedades e operações envolvidas, serão utilizadas regiões no plano

para representá-las. Primeiro, deve-se definir um ponto de referência, podendo ser interno ou externo ao item. A região é definida pelo conjunto de translações aplicadas ao ponto de referência (Fig. 3.1).

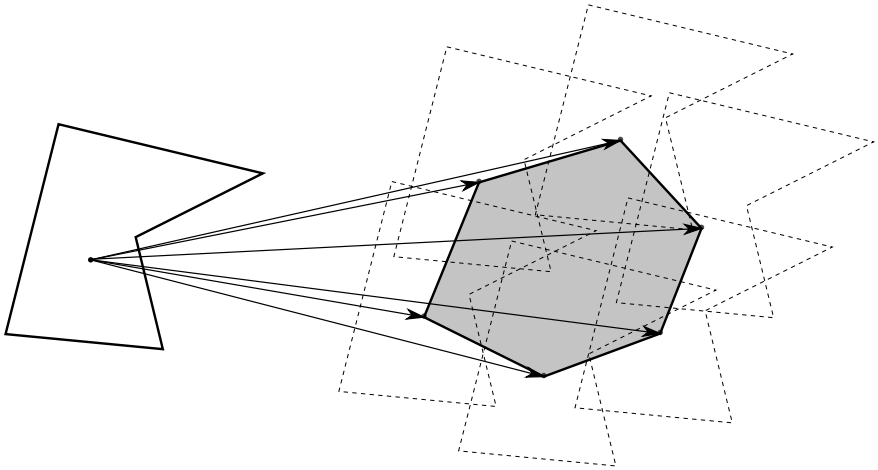


Figura 3.1: Conjunto de translações aplicadas a um item (polígono à esquerda) representado como uma região no espaço (região em cinza) formada por todas as translações do conjunto aplicadas a seu ponto de referência.

Considere dois itens.

Um item já está posicionado no espaço, denominado item fixo. Para o próximo item, móvel, existem translações que o sobrepõem ao item fixo. O polígono de obstrução é a região que representa o conjunto de translações proibidas para o item móvel, isto é, posições onde este item se intersecciona com o item fixo. Diz-se que o polígono de obstrução é induzido pelo item fixo ao item móvel. Um exemplo de polígono de obstrução pode ser visto na Fig. 3.2. Para um item P, considere $i(P)$ o seu interior, ∂P o seu contorno e $c(P)$ seu complemento.

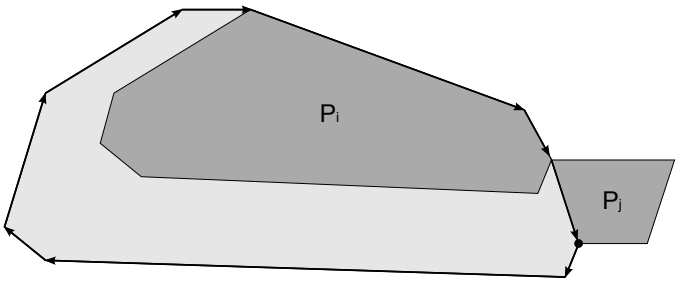


Figura 3.2: Polígono de obstrução (contorno definido pelas arestas orientadas) induzido pelo item P_i ao item item P_j. O ponto de referência coincide com o vértice inferior esquerdo do polígono P_j.

Definição 3.1 *Supondo dois polígonos P_i e P_j fixos. O polígono de obstrução induzido pelo item P_i ao item P_j, denotado por $\Upsilon(P_i, P_j)$, é o conjunto de vetores*

de translação que aplicado a P_j o levam a colidir com P_i. Desta forma,

$$\begin{aligned} \Upsilon(P_i, P_j) &= i(P_i) \ominus i(P_j) \\ &= \{\vec{\boldsymbol{v}} \,|\, \exists\, \boldsymbol{a} \in i(P_j), \boldsymbol{a} + \vec{\boldsymbol{v}} \in i(P_i)\} \end{aligned} \tag{3.1}$$

Diferentes algoritmos foram propostos para a determinação do polígono de obstrução. Mahadevan [29] desenvolveu um algoritmo baseado em um esquema de escorregamento. No entanto, algoritmos mais eficientes foram desenvolvidos posteriormente baseados na soma de Minkowski [12, 30]. Nestes algoritmos, foi constatado que é possível obter o polígono de obstrução utilizando a soma de Minkowski e o polígono oposto.

Definição 3.2 *A soma de Minkowski de dois polígonos P_i e P_j, denotada por $P_i \oplus P_j$, é definida como o conjunto de pontos $\{\boldsymbol{O} + \vec{\boldsymbol{v}} + \vec{\boldsymbol{w}} \,|\, \boldsymbol{O} + \vec{\boldsymbol{v}} \in P_i, \boldsymbol{O} + \vec{\boldsymbol{w}} \in P_j\}$.*

Definição 3.3 *Define-se polígono oposto a um dado polígono P_j denotado por $-P_j$ como o conjunto de pontos definido por $-P_j = \{\boldsymbol{O} - \vec{\boldsymbol{w}} \,|\, \boldsymbol{O} + \vec{\boldsymbol{w}} \in P_j\}$.*

Pode-se observar que, para obter o polígono oposto, só é necessário inverter o sinal de todas as coordenadas do polígono original.

A partir das definições 3.1–3.3, percebe-se que:

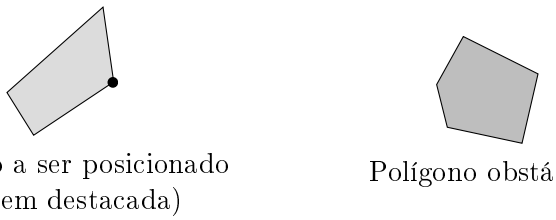
$$\Upsilon(P_i, P_j) = i(P_i) \ominus i(P_j) = i(P_i) \oplus (-i(P_j)) \tag{3.2}$$

ou seja, o polígono de obstrução é gerado pela soma de Minkowski do obstáculo com o oposto do polígono a ser posicionado. Definindo-se um ponto de referência do item P_i arbitrariamente, podendo ser externo ao polígono, as seguintes propriedades são válidas:

Propriedade 3.1 *O ponto de referência do item P_i pertencer a $\Upsilon(P_i, P_j) \Longrightarrow$ P_i e P_j se sobrepõem.*

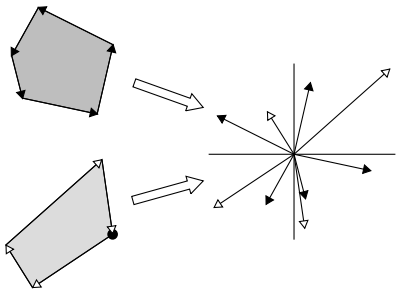
Propriedade 3.2 *O ponto de referência do item P_i pertencer a $\partial\Upsilon(P_i, P_j) \Longrightarrow$ P_i e P_j se encostam.*

Propriedade 3.3 *O ponto de referência do item P_i pertencer a $c(\Upsilon(P_i, P_j)) \Longrightarrow$ P_i e P_j estão separados.*

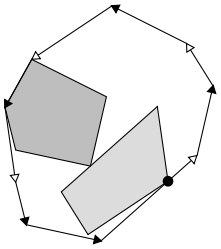


Polígono a ser posicionado
(origem destacada)

Polígono obstáculo



Extração das arestas dos polígonos
(nota-se a inversão nas arestas do polígono a ser posicionado)



Resultado: Polígono de obstrução gerado a partir das arestas dos dois polígonos

Figura 3.3: Exemplo de geração de polígono de obstrução para polígonos convexos.

Somas de Minkowski podem ser calculadas facilmente para polígonos convexos. O resultado da soma de Minkowski de dois polígonos convexos é um polígono convexo construído a partir das arestas dos polígonos originais (Fig. 3.3).

Polígonos não-convexos podem ser decompostos em uma etapa de pré-processamento do algoritmo, visto que as transformações aplicadas (de rotação e translação) não afetam tal decomposição. Sendo $\mathcal{D}$ o conjunto de polígonos convexos resultantes da decomposição do item côncavo P_c, o polígono de obstrução de P_c é dado pela união dos polígonos de obstrução induzidos a este item pelos polígonos do conjunto $\mathcal{D}$. Assim,

$$\Upsilon(P_i, P_c) = \bigcup_{P_k \in \mathcal{D}} \Upsilon(P_i, P_k) = \bigcup_{P_k \in \mathcal{D}} i(P_i) \ominus i(P_k) \tag{3.3}$$

O mesmo procedimento deve ser adotado caso o item fixo seja côncavo, ou seja:

$$\Upsilon(P_c, P_i) = \bigcup_{P_k \in \mathcal{D}} \Upsilon(P_k, P_i) = \bigcup_{P_k \in \mathcal{D}} i(P_k) \ominus i(P_i) \tag{3.4}$$

3.2 Polígono de posicionamento interno

O polígono de posicionamento interno (inner-fit polygon) é um conceito derivado do polígono de obstrução e representa o conjunto de translações possíveis para o posicionamento de um item dentro de um contêiner $\mathcal{C}$. O polígono de posicionamento interno pode ser computado através do deslizamento de um item contido no contêiner, ao longo de seu contorno [31] (Fig. 3.4).

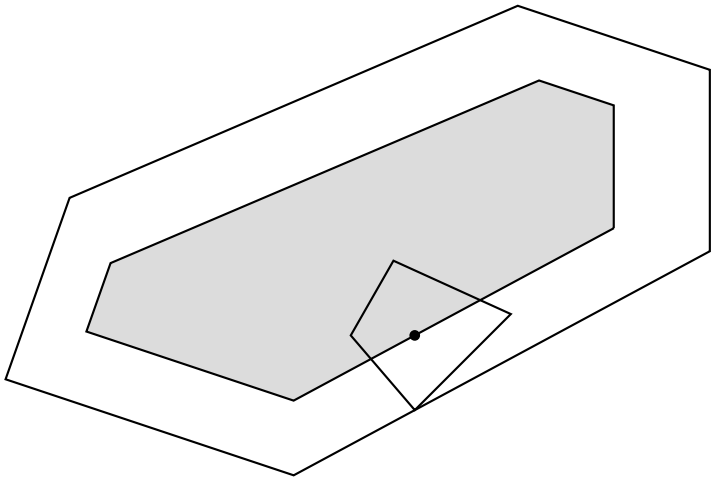


Figura 3.4: O polígono de posicionamento interno (em cinza) para um dado item e seu contêiner.

Definição 3.4 *O polígono de posicionamento interno induzido por um contêiner $\mathcal{C}$ ao item P_j, denotado por $\Lambda(\mathcal{C}, P_j)$, é o conjunto de vetores de translação que, aplicados a P_j, o posiciona dentro do contêiner. Segue,*

$$\begin{aligned} \Lambda(\mathcal{C}, P_i) &= c(c(\mathcal{C}) \oplus (-i(P_i))) \\ &= \{\vec{\boldsymbol{v}} \,|\, \forall\, \boldsymbol{a} \in i(P_i), \boldsymbol{a} + \vec{\boldsymbol{v}} \in \mathcal{C}\} \end{aligned} \tag{3.5}$$

As seguintes propriedades são válidas

Propriedade 3.4 *O ponto de referência do item P_i pertencer a $\Lambda(\mathcal{C}, P_i) \Longrightarrow P_i$ é interno a $\mathcal{C}$.*

Propriedade 3.5 *O ponto de referência do item P_i pertencer a $\partial\Lambda(\mathcal{C}, P_i) \Longrightarrow P_i$ é interno e encosta em $\mathcal{C}$.*

Propriedade 3.6 *O ponto de referência do item P_i pertencer a $c(\Lambda(\mathcal{C}, P_i)) \Longrightarrow$ Alguma região de P_i é externa a $\mathcal{C}$.*

3.3 Região livre de colisão

Considere um contêiner $\mathcal{C}$ e um conjunto de itens $\mathcal{P} = \{P_1, \cdots, P_n\}$ já posicionados, sem colisão, como pode ser visto na Fig. 3.5. Deseja-se inserir um novo item P_m, $m > n$ no interior do contêiner, sem sobreposição com os itens já posicionados. A região livre de colisão representa o conjunto de translações possíveis para o item P_m.

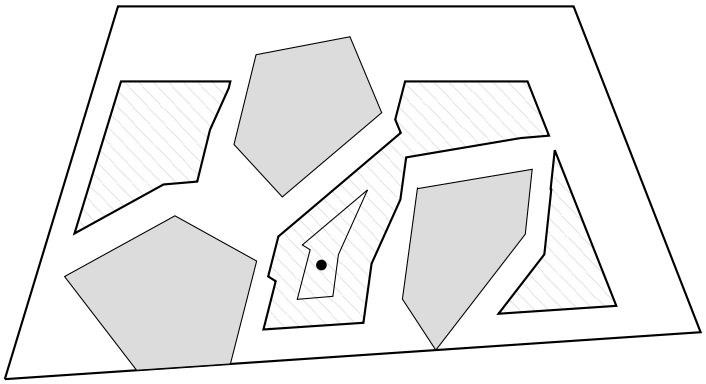


Figura 3.5: A região livre de obstrução (regiões hachuradas) para o posicionamento de um item em um contêiner com obstáculos (em cinza).

Definição 3.5 *A região livre de colisão é o conjunto de todas as translações que, aplicadas a um item específico, o posiciona no interior do contêiner sem que haja colisão com os itens já posicionados.*

Quando não existem itens posicionados, a região livre de colisão é o próprio polígono de posicionamento interno. Para um determinado item, o cálculo do polígono de posicionamento interno é o primeiro passo para se determinar a região livre de colisão. Na sequência, deve-se remover do polígono de posicionamento interno os polígonos de obstrução induzidos pelos itens já posicionados. Assim, a região livre de colisão, denotada por $\Pi(\mathcal{C}, \mathcal{P}, P_m)$, pode ser determinada através da expressão:

$$\Pi(\mathcal{C}, \mathcal{P}, P_m) = \Lambda(\mathcal{C}, P_m) - \bigcup_{P_i \in \mathcal{P}} i(P_i) \ominus i(P_m) \tag{3.6}$$

Deve-se notar que, uma vez que o polígono de obstrução representa apenas as translações proibidas, o seu contorno não deve ser considerado. O posicionamento

em seu contorno não resulta em colisão, conforme a propriedade 3.2. Já nos casos do polígono de posicionamento e da região livre de colisão, o contorno é uma região desejável, pois o posicionamento nesta região é válido. Desta forma, como o cálculo da região livre de colisão através da equação (3.6) envolve operações entre polígonos de posicionamento e de obstrução, é necessário um tratamento especial para desconsiderar os contornos dos polígonos de obstrução. Para isso, foram desenvolvidas operações Booleanas não regularizadas para resolver as equações que envolvem polígonos de obstrução (equações (3.3), (3.5) e (3.6)).

3.4 Paralelização da determinação da região livre de colisão

A equação (3.6) pode ser resolvida utilizando diferentes sequências de operações, sendo que a forma escolhida para resolvê-la influencia no desempenho do algoritmo. Neste trabalho, dois métodos para determinar a região livre de colisão foram adotados. O primeiro, método da subtração, subtrai cada polígono de obstrução do polígono de posicionamento interno. Desta forma, é realizada uma sequência de subtrações, como visto na Fig.3.6. O segundo método é constituído de duas etapas distintas. A primeira determina a união de todos os polígonos de obstrução e a segunda subtrai este resultado do polígono de posicionamento interno. Este método é ilustrado na Fig.3.7.

O método da subtração não pode ser trivialmente paralelizado. No segundo método, a paralelização é empregada para a primeira etapa, utilizando um computador com processador multinúcleo [32]. Para isso, é necessário inserir todos os polígonos de obstrução em uma fila. Cada núcleo retira dois polígonos de obstrução da fila e calcula o resultado. Depois, o resultado é reinserido na fila. Este processo é repetido $n-1$ vezes, considerando n o número total de polígonos de obstrução. Por fim, o resultado final das uniões é subtraído do polígono de posicionamento interno para se obter a região livre de colisão. O método paralelizado é empregado no exemplo da Fig. 3.8.

Através de análise do exemplo ilustrado nas Figs. 3.6 e 3.7 é possível observar que o desempenho do algoritmo serial deve ser diferente em cada caso. Isto se deve ao fato de que o número de vértices e de intersecções têm influência no tempo de execução do algoritmo. Assim, se contabilizarmos, para cada operação, o número de vértices dos dois polígonos e quantas vezes se interseccionam é possível obter uma estimativa de qual dos métodos terá menor tempo de execução. No caso da

utilização apenas de subtrações, em média, foram contabilizados $12,50$ vértices e $1,67$ intersecções. Já para o segundo método, foram observados $17,00$ vértices e $2,14$ intersecções em média por operação. Esta diferença resultará em tempos distintos de execução, como poderá ser observado nos resultados apresentados no capítulo 6.

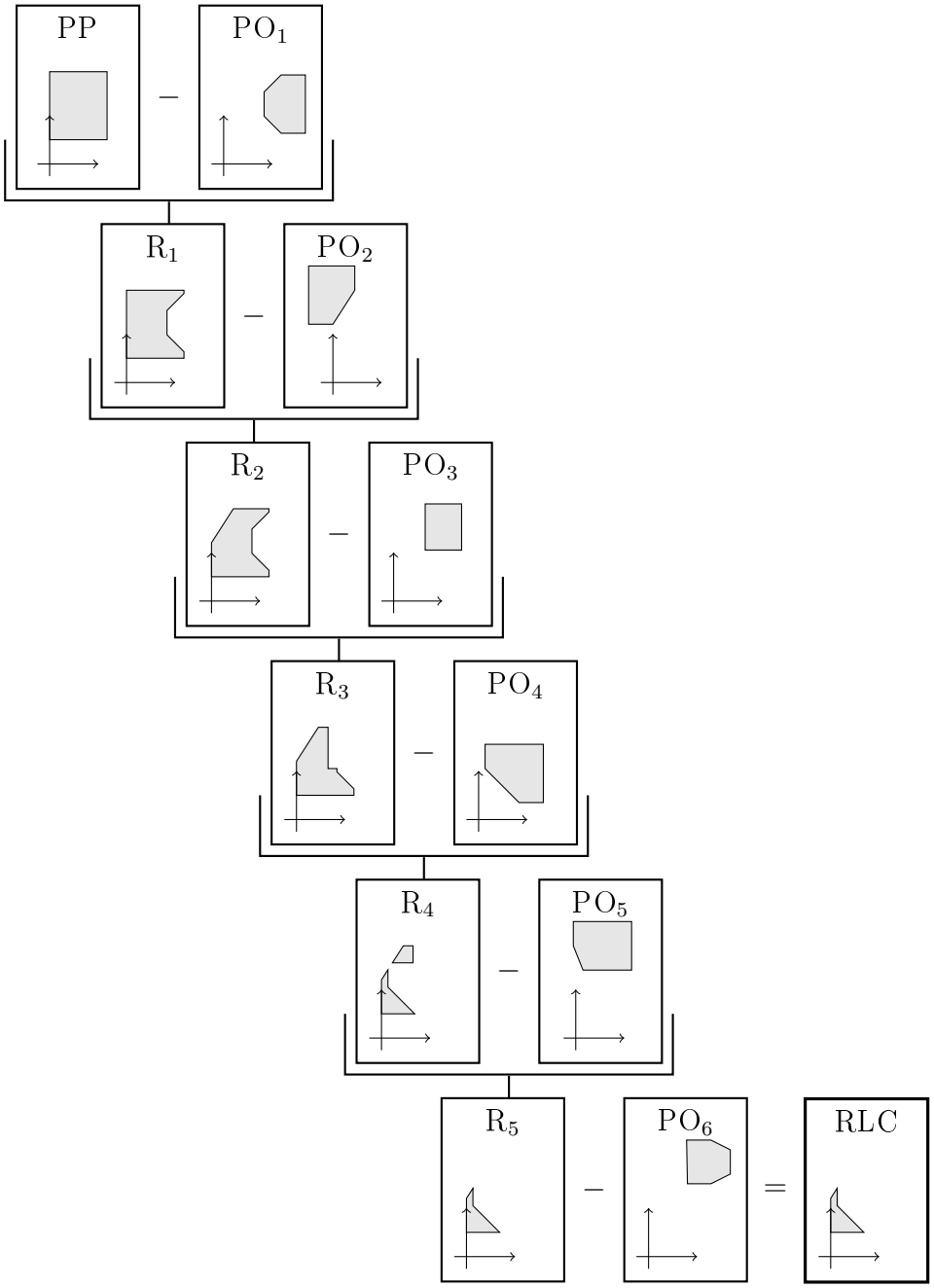


Figura 3.6: Exemplo de cálculo da região livre de colisão (RLC) utilizando apenas subtrações. Conjunto inicial de um polígono de posicionamento interno (PP) e seis polígonos de obstrução (PO_1-PO_6). R representa o resultado de uma operação.

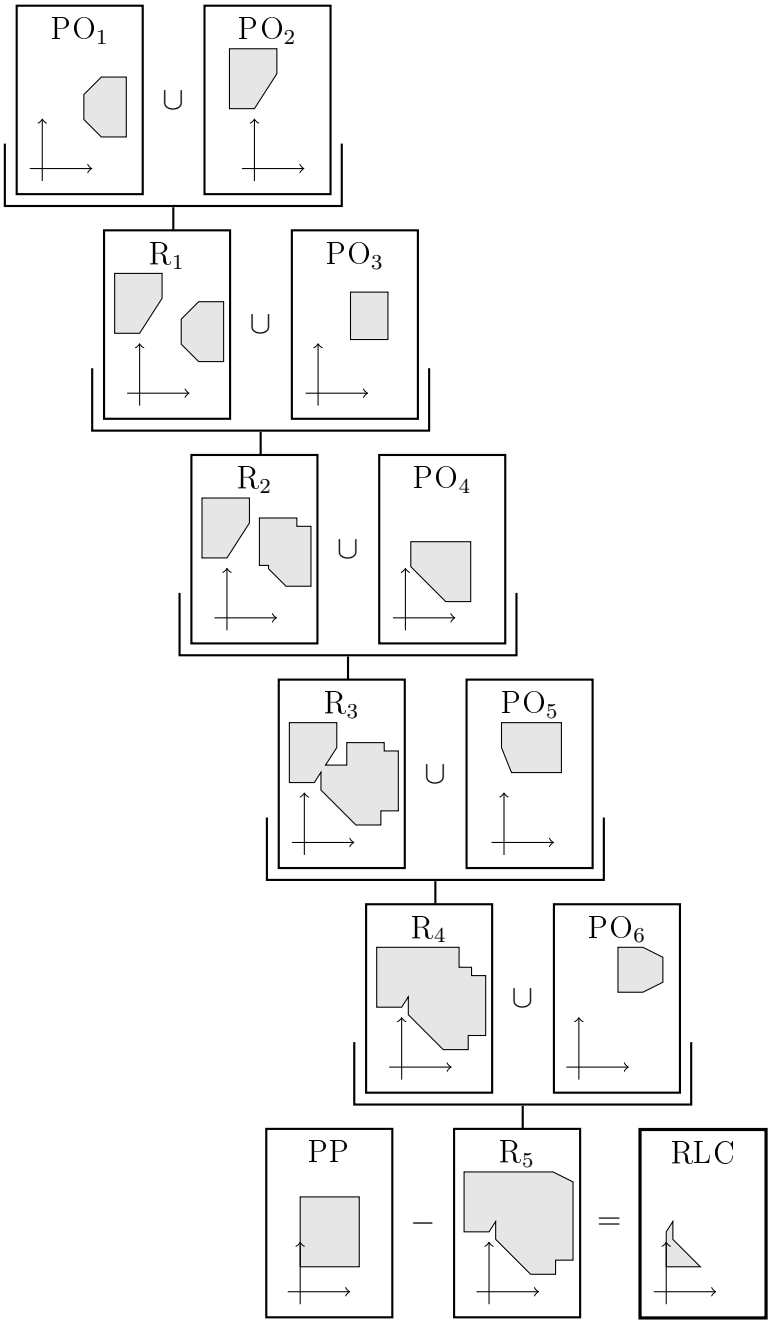


Figura 3.7: Exemplo de cálculo da região livre de colisão utilizando uniões e uma subtração. Os polígonos de obstrução de entrada e o polígono de posicionamento interno são os mesmos da Fig. 3.6. PO: Polígono de obstrução. R: Resultado da operação. PP: Polígono de posicionamento interno. RLC: Região livre de colisão.

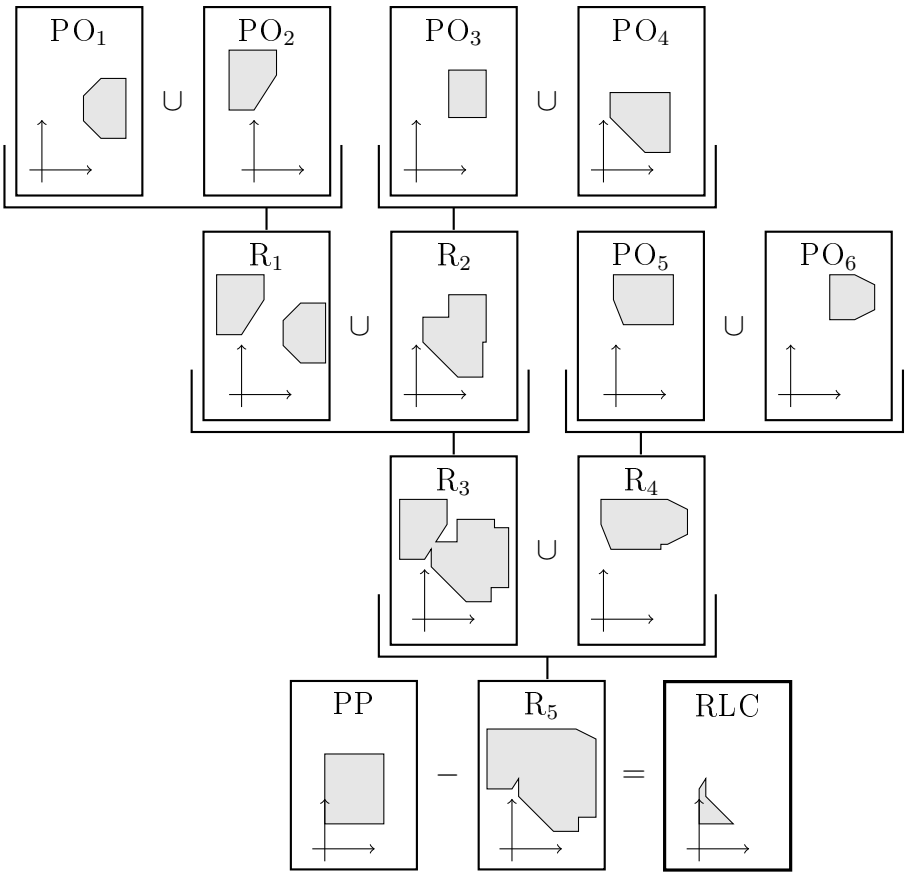


Figura 3.8: Exemplo de cálculo da região livre de colisão utilizando uniões e uma subtração em paralelo executado em dois núcleos. Os polígonos de obstrução de entrada e o polígono de posicionamento interno são os mesmos da Fig. 3.6. PO: Polígono de obstrução. R: Resultado da operação. PP: Polígono de posicionamento interno. RLC: Região livre de colisão.

4 Operações Booleanas não regularizadas

Para o cálculo da região livre de colisão foram implementadas operações Booleanas não regularizadas. Este capítulo descreve o algoritmo de operações Booleanas não regularizadas desenvolvido. Modificações foram realizadas a partir de um algoritmo de operações Booleanas regularizadas. Este algoritmo foi utilizado como base e as alterações foram realizadas nos casos específicos em que o resultado ou os polígonos de entrada são degenerados. Cada caso é descrito e o resultado correto correspondente é apresentado.

O algoritmo adotado possui precisão finita. Desta forma, as operações são realizadas em um reticulado. Consiste essencialmente em três etapas: a determinação das intersecções das arestas, a classificação e a coleta das arestas. A primeira etapa utiliza o algoritmo desenvolvido por Hobby [33] para identificar e processar as intersecções em um reticulado.

A seção 4.1 é uma introdução para as operações Booleanas não regularizadas. As seguintes seções 4.2–4.5 descrevem as etapas do algoritmo: determinação de intersecções, classificação e coleta de arestas, incluindo as novas regras para adaptar o algoritmo de operações Booleanas regularizadas para não regularizadas. Por fim, na seção 4.7 são discutidas as medidas tomadas para a implementação do algoritmo proposto.

4.1 Introdução

As operações Booleanas regularizadas assumem que o contorno pertence ao polígono, ao passo que em operações não regularizadas é possível considerar apenas a região interna. Na Fig. 4.1 pode-se observar um caso crítico em que quatro itens retangulares já foram posicionados, e um quinto item retangular deve ser inserido. Verifica-se que o resultado correto somente é obtido quanto é considerada apenas a região interna dos polígonos de obstrução. Isto provém da própria definição do

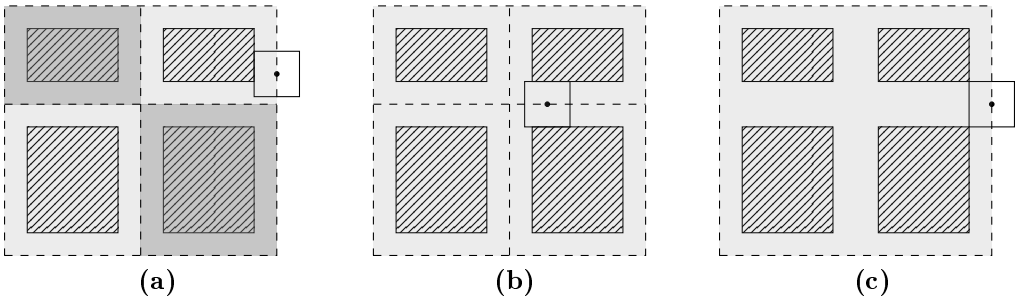


Figura 4.1: Exemplo de configuração onde foram posicionados quatro itens retangulares hachurados. O item retangular com o ponto de referência no centro é móvel. (a) Os quatro polígonos de obstrução determinados para o posicionamento do item. (b) A região de obstrução real, possuindo linhas internas que permitem o posicionamento do item retangular entre dois itens retangulares já posicionados. (c) Resultado da união regularizada dos quatro polígonos de obstrução, que não possui linhas internas.

polígono de obstrução. Assim, o único meio de se obter a região livre de colisão corretamente é utilizar operações Booleanas não regularizadas [34]. No caso da Fig. 4.1 também é possível observar que o posicionamento da nova peça nas linhas internas é o com melhor compactação. Estes pontos só podem ser encontrados se adotadas operações Booleanas não regularizadas.

Assim como no caso descrito, a utilização de operações Booleanas não regularizadas pode resultar em regiões com arestas ou vértices desconexos, que não fazem parte do contorno de um polígono. Estes serão chamados de arestas e vértices salientes. Quando não for especificado, a aresta ou vértice pertence a um contorno. O posicionamento nos vértices ou arestas salientes geralmente proporcionam maior compactação.

Existem trabalhos sobre operações Booleanas não regularizadas [35, 36], no entanto não implementam corretamente o cálculo da região livre de colisão. Entretanto, para este trabalho, foi desenvolvido um novo algoritmo baseado em operações Booleanas regularizadas, uma vez que a sua aplicação é muito específica.

As propostas de operações Booleanas regularizadas existentes [37–39] possuem em comum a divisão em três passos: a determinação das intersecções entre arestas, a classificação das arestas e a seleção das arestas apropriadas para compor o resultado final. O algoritmo proposto em [40] foi escolhido como base, pois é um algoritmo robusto e apresenta desempenho superior aos demais. Este algoritmo trabalha com precisão finita, que exige atenção especial na etapa de determinação das intersecções.

4.2 Detecção de intersecções

A detecção de intersecções é a primeira etapa para realizar a operação Booleana entre polígonos. Uma forma direta de realizar esta etapa é verificar todos os pares de segmentos e armazenar as intersecções à medida que são encontradas. No entanto, existem métodos mais eficientes, que realizam a checagem de um número menor que o total de pares. Um exemplo deste tipo de abordagem foi proposto por Bentley e Ottmann [41], que é baseado em Shamos e Hoey [42], cujo algoritmo é adotado neste trabalho.

O algoritmo de Shamos e Hoey [42] é capaz de reportar se existem dois segmentos que se interseccionam em um grupo de N segmentos. Ambos algoritmos adotam uma abordagem que se utiliza de uma linha vertical imaginária, linha de varredura (sweep line), que percorre a área contendo todos os segmentos de reta. É definida uma lista de todos os segmentos que cruzam a linha de varredura. Esta lista deve estar sempre ordenada em relação à coordenada vertical do ponto de intersecção da linha de varredura com o segmento (Fig. 4.2). Se nenhuma modificação ocorre na lista durante todo percurso da linha de varredura, podemos afirmar que não há nenhuma intersecção no conjunto de segmentos. Entretanto, caso seja detectada alguma modificação na lista deve-se verificar os segmentos que podem se interseccionar. A lista só é modificada na ocorrência de três eventos: início de segmento, fim de segmento ou intersecção de segmentos. A Fig. 4.3 descreve quando cada um dos três eventos ocorre.

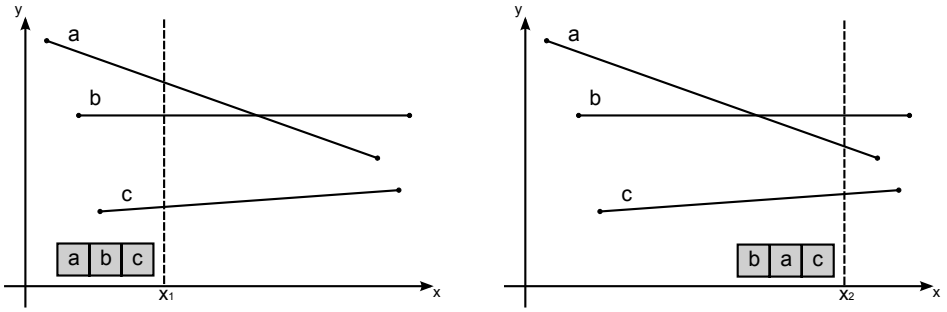


Figura 4.2: Exemplo de utilização da linha de varredura. As caixas em cinza representam a lista ordenada com os três segmentos para a linha de varredura (linha tracejada) em duas posições distintas (x_1 e x_2).

Desta forma, o cálculo para a investigação do ponto de intersecção só deve ser realizado para os pares de segmentos escolhidos de acordo com as seguintes regras:

- Para o evento de início de segmento: Após inserir o segmento novo na lista, deve-se investigar a intersecção do novo segmento com os segmentos

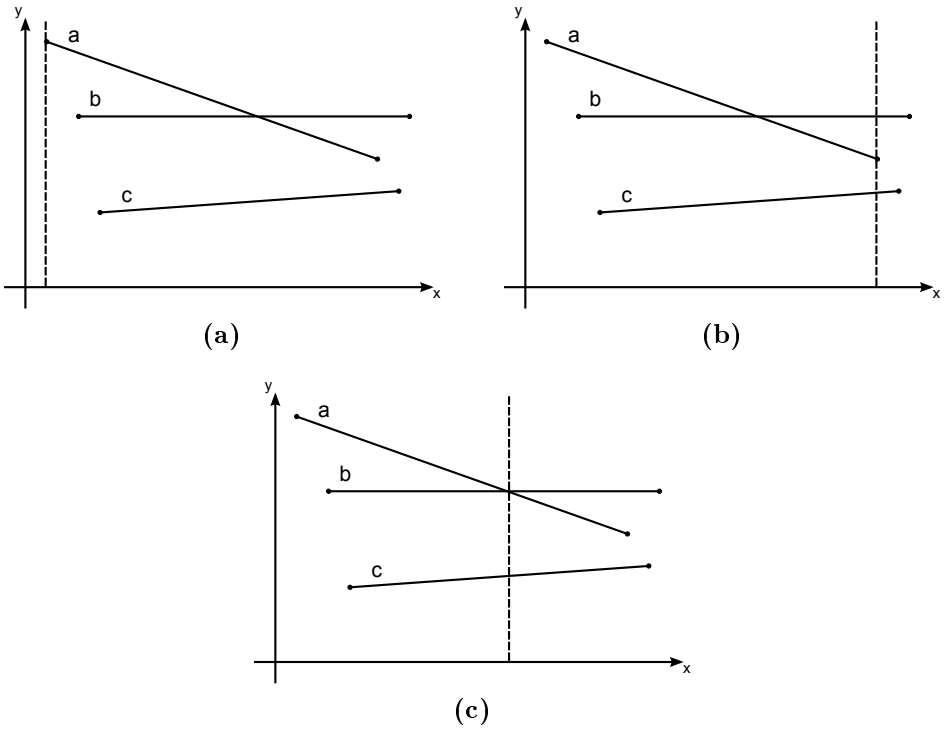


Figura 4.3: Exemplo da ocorrência dos três tipos de eventos. A linha de varredura está representada por uma linha tracejada. (a) Início de segmento. (b) Fim de segmento. (c) Intersecção de segmentos.

imediatamente superior e inferior.

- Para o evento de fim de segmento: Após remover o segmento novo na lista, deve-se investigar se os segmentos imediatamente superior e inferior se interseccionam.

- Para o evento de intersecção: Inverter a ordem dos segmentos que se interseccionam. Após a inversão, definimos estes segmentos como a e b, sendo que a está acima de b. Deve-se investigar a intersecção do segmento a com o seu imediatamente acima, e do segmento b com o seu imediatamente abaixo.

É importante observar que os eventos de início e fim de segmento são conhecidos a priori, enquanto o evento de intersecção é detectado durante a execução do algoritmo. Estes eventos são mantidos na forma de uma lista. Assim, cada vez que uma intersecção é detectada, deve-se incluir o evento correspondente na lista de eventos.

O algoritmo descrito possui a complexidade de $O(NlogN + klogN)$, sendo N o número total de segmentos e k a quantidade de intersecções existentes no conjunto. A complexidade de se comparar todos os pares de segmentos para encontrar as intersecções é $O(N^2)$.

4.2.1 Inserindo os pontos de intersecção

O algoritmo de intersecção utilizado por [41] foi desenvolvido para aritmética real exata. No entanto, foi adotada precisão finita para o problema estudado. Por esta razão, a utilização direta das intersecções encontradas pelo algoritmo de Bentley e Ottmann [41] não é adequada. O principal erro que pode ocorrer ao se utilizar as coordenadas aproximadas das intersecções é o surgimento de novas intersecções, como pode ser visto na Fig. 4.4. Outro problema é lidar com segmentos verticais e intersecção de três segmentos em um mesmo ponto. Para resolver estes problemas, foi utilizado um algoritmo baseado em [33].

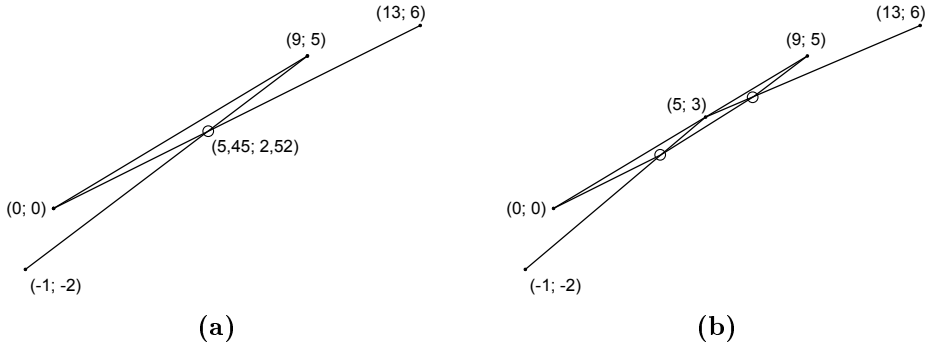


Figura 4.4: Exemplo retirado de Hobby [33]. (a) Segmentos de entrada. (b) Resultado obtido utilizando aproximação das coordenadas da intersecção. O surgimento de novas intersecções estão indicados por círculos.

O algoritmo se utiliza das informações colhidas durante a etapa da determinação das intersecções. Quando a linha de varredura encontra um evento de intersecção, é adicionado um novo passo, que consiste na busca pelos segmentos que estejam próximos à intersecção, e inserção de um vértice para cada segmento encontrado. Os segmentos são escolhidos definindo um quadrado de tolerância em torno do ponto de intersecção, já com as coordenadas ajustadas para a precisão utilizada, e verificando todos os segmentos que cruzam este quadrado.

As medidas tomadas para resolver os problemas de entradas com segmentos na horizontal ou intersecção de três segmentos em um mesmo ponto podem ser verificadas em Hobby [33]. A descrição dessas medidas não é importante para o escopo deste trabalho.

4.3 Classificação de arestas e contornos

Considere uma operação Booleana entre duas regiões A e B. Regiões podem ser compostas de mais de um polígono e apresentar arestas ou vértices degenerados. Na etapa de classificação são aplicados rótulos em ambas as regiões. Neste texto,

apenas a classificação da região A será descrita. Desta forma, os rótulos indicam a posição da aresta ou contorno em relação ao contorno de B. Para realizar a classificação de B, os mesmos procedimentos devem ser realizados, desta vez considerando as posições em relação à região A.

A etapa de classificação é realizada em dois níveis. No primeiro classifica-se o contorno, e no segundo as arestas. O contorno pode ser interno, externo ou interseccionar a região B. O segundo nível só deverá ser realizado caso o contorno intersecione a região B. Para as arestas, considera-se que a determinação e inserção das intersecções foram realizadas corretamente. Desta forma, pode-se concluir que qualquer aresta não coincidente ou se intersecciona nos seus pontos extremos ou não se intersecciona. Sendo assim, cada aresta de A pode ser coincidente com uma aresta de B, ou estar no interior ou exterior de B.

Também deve-se classificar as arestas e vértices salientes. As arestas salientes são rotuladas seguindo o mesmo procedimento adotado para as arestas de contorno, desconsiderando a orientação. Os vértices salientes de A podem se situar no interior, exterior ou sobre a região B, ou seja, sobre uma aresta de contorno, uma aresta saliente, ou coincidente com um vértice saliente de B. Os rótulos definidos a partir dessas observações podem ser vistos na Tabela 4.1.

Tabela 4.1: Rótulos para a região A. Elementos salientes são arestas e vértices salientes

Tipo	Rótulo	Descrição
Contorno	INTERNO	Está no interior de B
	EXTERNO	Está no exterior de B
	INTERSECTA	Cruza com B
Aresta	INTERNA	Está no interior de B
	EXTERNA	Está no exterior de B
	COMPARTILHADA1	Coincide com uma aresta de B com a mesma direção
	COMPARTILHADA2	Coincide com uma aresta de B com a direção oposta
Elementos Salientes	INTERNO	Está no interior de B
	EXTERNO	Está no exterior de B
	COMPARTILHADO	Vértice saliente está sobre a região B Aresta saliente coincide com uma aresta (de contorno ou saliente) de B

A Fig. 4.5 apresenta um exemplo de classificação em que é possível identificar todos os rótulos para arestas, arestas salientes e vértices salientes.

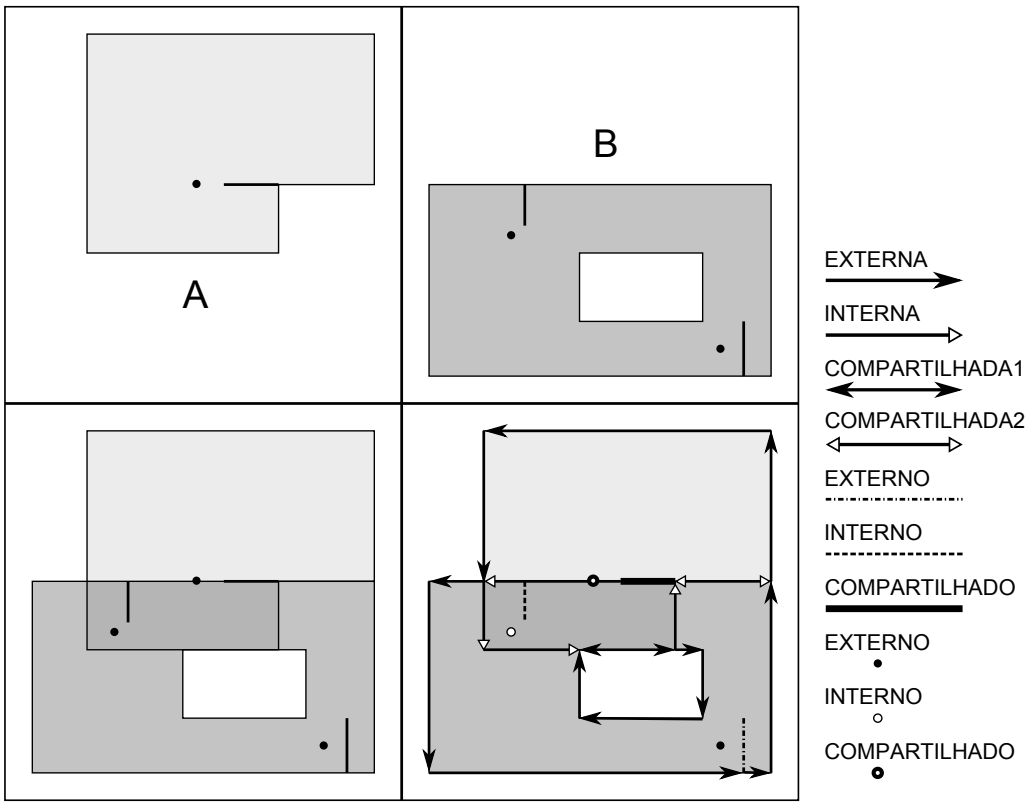


Figura 4.5: Exemplo de dois poligonos A e B classificados segundos os rótulos definidos na Tabela 4.1.

4.4 Coleta de arestas e contornos

Leonov e Nikitin [40] mostram que, para se obter os contornos finais resultantes, é necessário decidir quais arestas incluir e sua direção, seguindo algum contorno de uma região de entrada e aplicando regras caso encontre alguma intersecção. A coleta consiste em incluir na região resultante arestas de forma ordenada com a finalidade de se obter contornos. Para o caso de operações não regularizadas, a coleta possui mais dois objetivos: determinar se há geração de novos elementos salientes, arestas ou vértices, no resultado, e decidir quais elementos salientes presentes nas regiões de entrada devem ser mantidos ou removidos.

Assim como na classificação, a coleta é realizada primeiro nos contornos e, caso um deles tenha sido classificado como INTERSECTA, deve-se realizar a coleta de cada aresta deste contorno.

Os contornos resultantes da operação Booleana regularizada são os mesmos obtidos pela operação não regularizadas. Por isso, a coleta dos contornos e arestas são as mesmas. No entanto, uma operação realizada entre regiões sem arestas ou vértices salientes pode resultar em uma região com arestas ou vértices salientes. A verificação de geração de novos vértices ou arestas salientes deve ser feita durante a coleta dos contornos. O processamento de arestas e vértices salientes já

existentes nos polígonos de entrada na coleta pode ser realizado separadamente dos contornos.

É importante ressaltar que as operações não regularizadas desenvolvidas são específicas para resolver a equação (3.6). Desta forma, as únicas operações necessárias são: a união de polígonos de obstrução e a subtração com o minuendo sendo uma região livre de colisão e o subtraendo um polígono de obstrução. Sendo assim, considerando-se as propriedades das entidades envolvidas, a operação de união considera que cada um dos polígonos de entrada não possui o seu contorno, resultando em um polígono que também não possui contorno. Já no caso da subtração, deve-se considerar o contorno do minuendo e desconsiderar o do subtraendo. O resultado da operação é um polígono que possui contorno.

São analisados todos os casos possíveis para a geração e coleta de arestas e vértices salientes para as operações de união e subtração não regularizadas. A partir destas análises um novo conjunto de regras é definido para cada uma das operações.

4.4.1 Estudo sobre a operação de união

Os casos podem ser divididos em três grupos: geração de arestas e vértices salientes, coleta de arestas salientes e coleta de vértices salientes.

- Geração de arestas e vértices salientes – casos em que o resultado da união apresenta a inclusão de arestas ou vértices salientes que não existiam nas regiões de entrada da operação. Neste grupo existem três casos. O primeiro e o segundo apresentam a geração de arestas e vértices salientes, Fig. 4.6a e Fig. 4.6b respectivamente, a partir de regiões sem elementos salientes. O caso da Fig. 4.6c representa a geração de um vértice saliente a partir do cruzamento de duas arestas salientes.

- Coleta de arestas salientes – casos em que a manutenção ou exclusão de uma aresta saliente determina o resultado correto da operação. Os dois primeiros casos desse grupo apresentam o resultado em que a aresta saliente é classificada como EXTERNO (Fig. 4.7a) ou INTERNO (Fig. 4.7b). Os outros dois casos, ilustrados nas Fig. 4.7c e Fig. 4.7d, mostram o resultado quando a aresta saliente coincide com um contorno ou com outra aresta saliente, respectivamente.

- Coleta de vértices salientes – casos de manutenção ou exclusão dos vértices salientes das regiões de entrada. Como no grupo de coleta de arestas

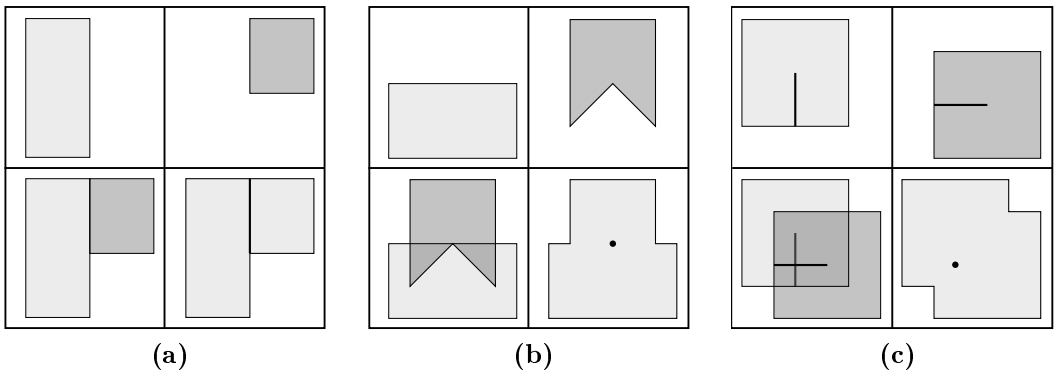


Figura 4.6: Casos do grupo de geração de elementos salientes para a união Booleana não regularizada. Considere a operação $A \cup B$. A figura do canto superior esquerdo é a região A. A figura do canto superior direito é a região B. A figura do canto inferior esquerdo é a superposição das regiões A e B. A figura do canto inferior direito é o resultado da operação.

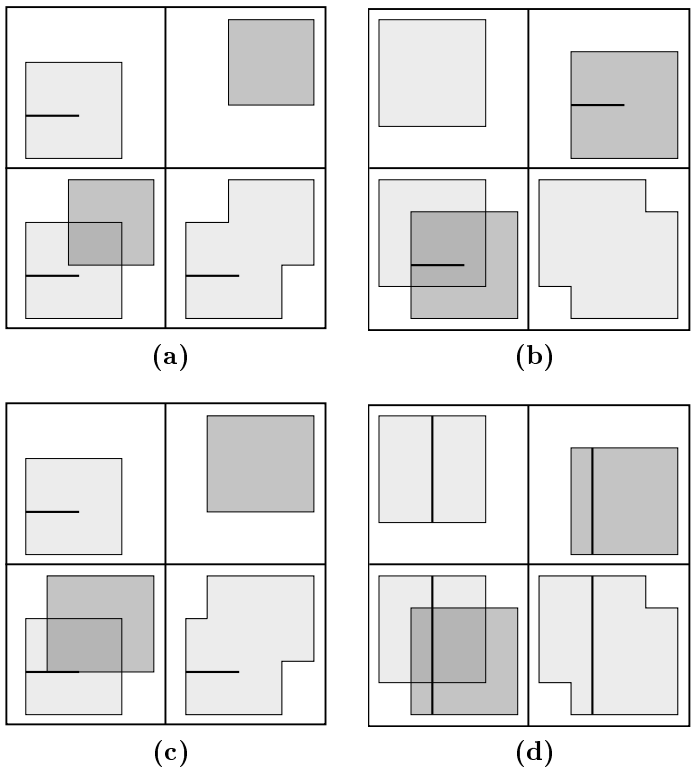


Figura 4.7: Casos do grupo de coleta de arestas salientes para a união Booleana não regularizada.

salientes, ocorrem situações em que o vértice é EXTERNO (Fig. 4.8a) ou INTERNO (Fig. 4.8b). Os dois casos seguintes são os quais o vértice saliente se encontra sobre do contorno do polígono, como visto na Fig. 4.8c, ou sobre uma aresta saliente (Fig. 4.8d). Por fim, um último caso deve ser considerado, o qual o vértice saliente coincide com outro, caso ilustrado na Fig. 4.8e.

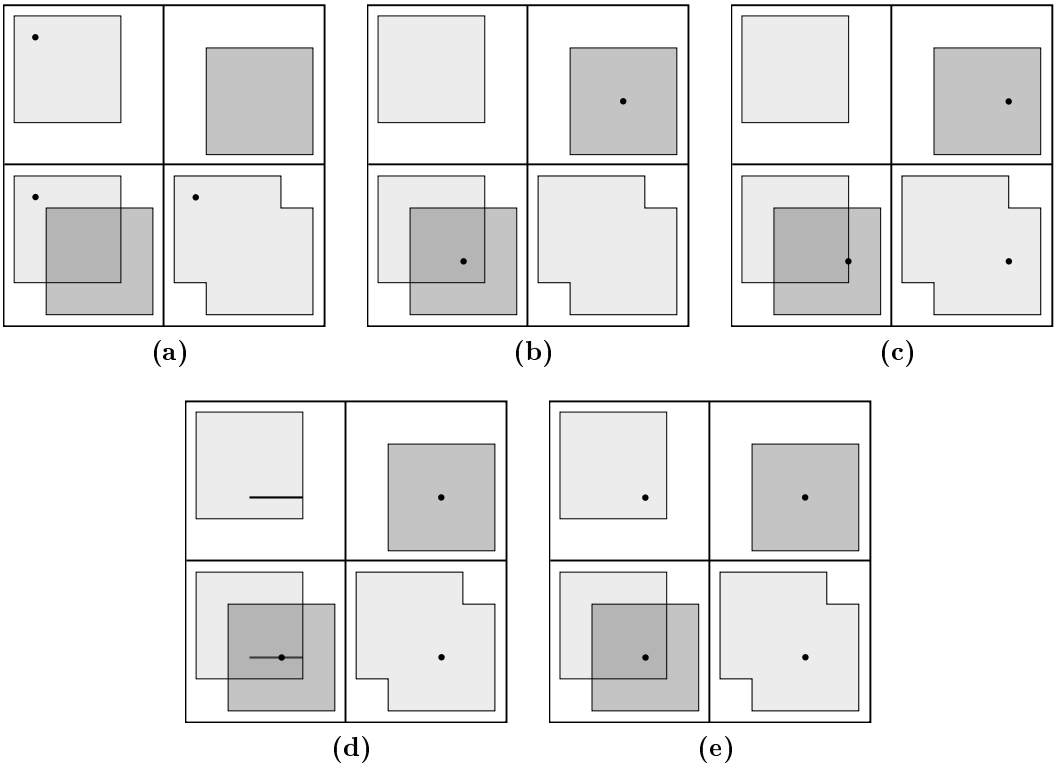


Figura 4.8: Casos do grupo de coleta de vértices salientes para a união Booleana não regularizada.

Tendo em vista todos os casos abordados, é possível estabelecer um conjunto de regras complementares para a operação de união não regularizada. Quando as arestas salientes forem classificadas como EXTERNO ou COMPARTILHADO, devem ser mantidas no resultado. As regras para vértices salientes são idênticas, como esperado, uma vez que possuem as mesmas propriedades que as arestas salientes. Também é possível concluir que não há diferença no resultado quando a aresta coincide com o contorno ou com outra aresta saliente. Para os vértices salientes, o resultado é o mesmo quando se encontram sobre uma aresta de contorno ou sobre uma aresta ou vértice salientes.

A geração de arestas salientes é simples, pois apenas ocorre no caso COMPARTILHADA2. No entanto, a geração de vértice saliente é mais complexa pois difere de todas as outras regras. Para a coleta de arestas (salientes ou não) só é necessário verificar o rótulo aplicado. Entretanto, a geração de vértice saliente

ocorre a partir de um vértice de contorno, que não possui rótulo. Aplicar rótulos em vértices de contorno não é solução para o problema, pois a geração de vértices salientes é dependente da posição relativa das arestas.

Para resolver o problema, foi utilizado o conceito de vértice de cruzamento (cross-vertex) apresentada em [40]. O vértices de cruzamento são vértices que correspondem aos pontos de intersecção. Em cada ponto de intersecção existem no mínimo dois vértices de cruzamento. É possível observar que a geração de um vértice saliente ocorrerá apenas em um vértice de cruzamento. Mais especificamente, quando todas as arestas que possuem vértice de cruzamento de um mesmo ponto de intersecção são internas a algum contorno da região, isto é, são classificadas como INTERNO ou são arestas salientes.

4.4.2 Estudo sobre a operação de subtração

Na subtração devemos analisar os mesmos casos que foram reportados na união não regularizada. A Fig. 4.9a apresenta o caso de geração de aresta saliente e as Fig.4.9b e 4.9c mostram a geração de vértices salientes a partir de intersecção de contornos e de arestas salientes, respectivamente. A coleta de arestas salientes estão representadas nas Fig. 4.10a–4.10d e a coleta de vértices nas Fig. 4.11a–4.11e.

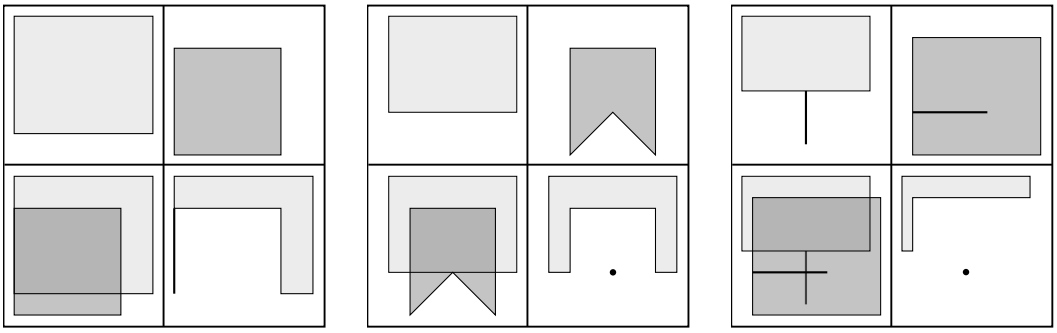


Figura 4.9: Casos do grupo de geração de elementos salientes para a subtração Booleana não regularizada.Considere a operação $A-B$. A figura do canto superior esquerdo é a região A. A figura do canto superior direito é a região B. A figura do canto inferior esquerdo é a superposição das regiões A e B. A figura do canto inferior direito é o resultado da operação.

Para desenvolver as regras da subtração, considere novamente a operação $A - B$. A coleta de arestas ou vértices salientes obedecem as mesmas regras, como na união. As regras de inclusão do elemento saliente são as seguintes:

1. Deve pertencer à região A e classificado como EXTERNO, ou

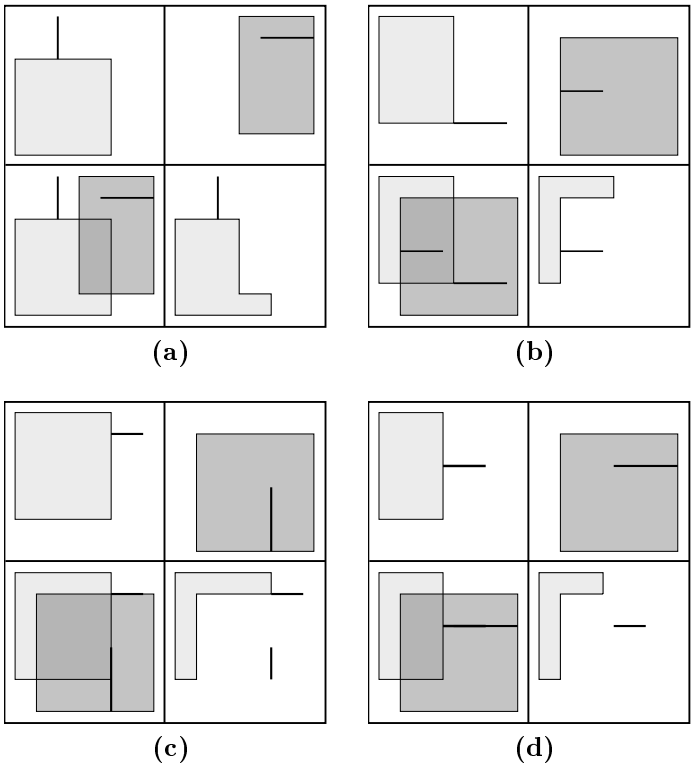


Figura 4.10: Casos do grupo de coleta de arestas salientes para a subtração Booleana não regularizada.

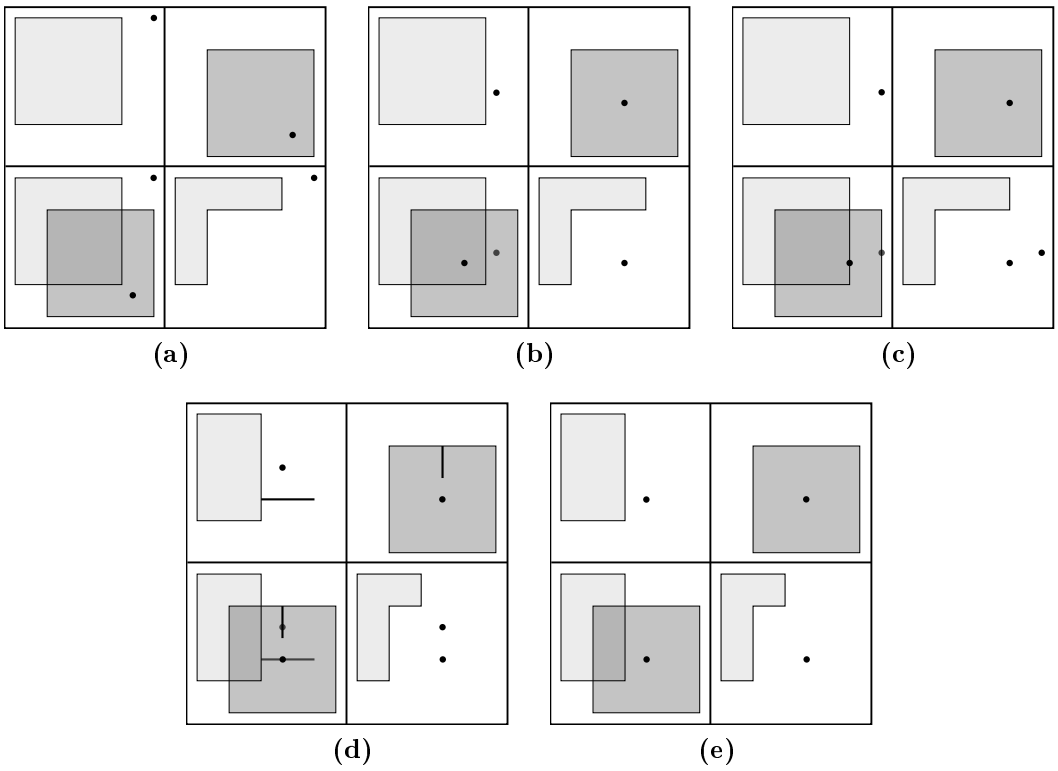


Figura 4.11: Casos do grupo de coleta de vértices salientes para a subtração Booleana não regularizada.

2. pertencer à região B e classificado como INTERNO, ou

3. pertencer à qualquer região e possuir a classificação de COMPARTILHADO.

A geração de arestas salientes ocorre quando se detecta uma arestas do tipo COMPARTILHADA1. Assim como na união, para se determinar a geração de um vértice saliente, é necessário analisar as arestas que possuem algum vértice de cruzamento. Dada uma intersecção, verificar o conjunto de arestas que possuam um vértice de cruzamento correspondente. Para que um vértice saliente seja gerado nesta coordenada, cada aresta do conjunto:

1. Deve pertencer à região A e classificada como INTERNO, ou

2. pertencer à região B e classificado como EXTERNA.

Caso alguma das arestas do conjunto não obedeça às regras, o vértice de cruzamento não deve ser inserido no resultado.

4.5 Regras da coleta para as duas operações

O estudo das operações de união e subtração não regularizadas resultou em um conjunto de regras para tratar os vértices e arestas salientes. A coleta de arestas e vértices salientes obedece às mesmas regras definidas para as arestas de contorno, definidas em [40]. Como os elementos salientes não possuem orientação, na regra de inclusão para estes elementos os rótulos COMPARTILHADO1 e COMPARTILHADA2 se tornam apenas COMPARTILHADO.

A geração de aresta saliente ocorre apenas quando existirem arestas compartilhadas. Existem dois tipos de compartilhamento, sendo que para as duas operações um tipo de compartilhamento constitui uma condição suficiente para inclusão no contorno do resultado. Sendo assim, apenas o tipo de compartilhamento que não consta nas regras de inclusão de aresta de contorno deve ocasionar a geração de uma aresta saliente. Para a união, é o caso do COMPARTILHADA2. Já para a subtração não regularizada, uma aresta COMPARTILHADA1 gera uma aresta saliente no resultado.

É possível constatar que, para os casos de união e subtração não regularizados, algum vértice de cruzamento de uma dada intersecção é sempre inserido no resultado. No resultado, este vértice de cruzamento pode ser um vértice do

contorno, o ponto inicial ou final de uma aresta saliente ou um vértice saliente. Também observa-se que a geração de vértices salientes só pode ocorrer onde se localizam os vértices de cruzamento. Desta forma, o único caso em que há geração de vértice saliente ocorre quando nenhuma das arestas que possuem um vértice no ponto de intersecção é mantida. Esta conclusão é coerente com as regras elaboradas para geração de vértices salientes para a união e subtração não regularizadas.

A Tabela 4.2 apresenta todas as regras adicionais para realizar a coleta nas operações não regularizadas de união e subtração.

Tabela 4.2: Regras para a coleta nas operações de união e subtração não regularizadas. Operação entre regiões A e B, sendo $A \cup B$ para união e $A - B$ para subtração

Operação	Elemento	Região	Regra ou rótulo	Ação
União	Vértice ou Aresta Salientes	A ou B	EXTERNO ou COMPARTILHADO	Inclusão
	Aresta	A ou B	COMPARTILHADA2	Geração
Subtração	Vértice ou Aresta Salientes	A B A ou B	EXTERNO INTERNO COMPARTILHADO	Inclusão
	Aresta	A ou B	COMPARTILHADA1	Geração
União ou Subtração	Vértice	A ou B	Intersecção onde nenhuma das arestas obedece a regra de inclusão	Geração

4.6 Representação de Polígonos

Para realizar as operações Booleanas, a representação adotada em [40] foi escolhida. Nela, os polígonos são representados por uma lista de vértices duplamente ligada. Os vértices são ordenados de acordo com a orientação das arestas e o primeiro vértice da lista é sempre o superior mais à esquerda. Um exemplo de lista de vértices que representa um polígono pode ser visto na Fig. 4.12.

Em operações Booleanas entre polígonos é comum o resultado possuir mais de um polígono (como no exemplo da Fig. 4.13). Por este motivo, é utilizado o conceito de região, capaz de representar um conjunto de polígonos. As operações Booleanas ocorrem entre duas regiões, resultando em uma nova região.

Os polígonos de uma região podem possuir orientação inversa, constituindo

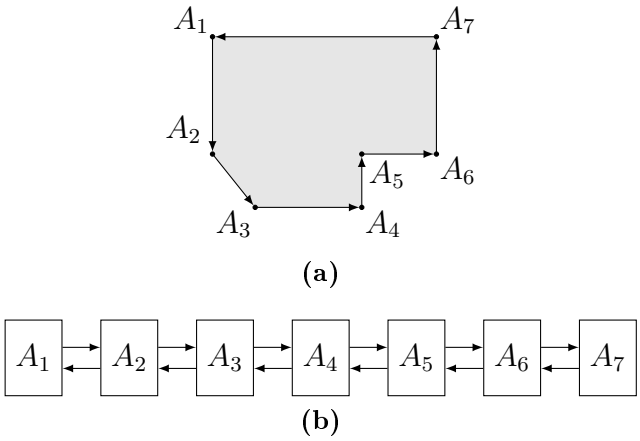


Figura 4.12: Exemplo de representação de um polígono. (a) Polígono adotado, com os vértices indexados. (b) Lista de vértices que representa o polígono.

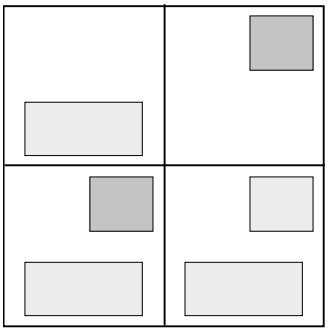


Figura 4.13: Casos em que a união de polígonos resulta em dois polígonos. Considere a operação $A \cup B$. A figura do canto superior esquerdo é a região A. A figura do canto superior direito é a região B. A figura do canto inferior esquerdo é a superposição das regiões A e B. A figura do canto inferior direito é o resultado da operação.

furos, como no exemplo da Fig. 4.14. Cada região é representada por uma lista duplamente ligada de polígonos, que por sua vez são representados por uma lista de vértices. Utilizando este novo conceito é possível manter a consistência das operações Booleanas.

4.7 Implementação

Foram elaboradas as regras que devem ser aplicadas na etapa de coleta para arestas e vértices salientes, além da geração dos mesmos. No entanto, a inclusão desses elementos implica em mudanças em outras etapas do algoritmo para que o algoritmo funcione corretamente. Nesta seção serão discutidos estes problemas mais específicos para implementação de um algoritmo de operações Booleanas não regularizadas baseado em [40]. A detecção de intersecções, primeira etapa do algoritmo, é realizada apenas considerando-se segmentos de retas. Desta forma, ao incluir arestas salientes na entrada do algoritmo, o resultado permanecerá

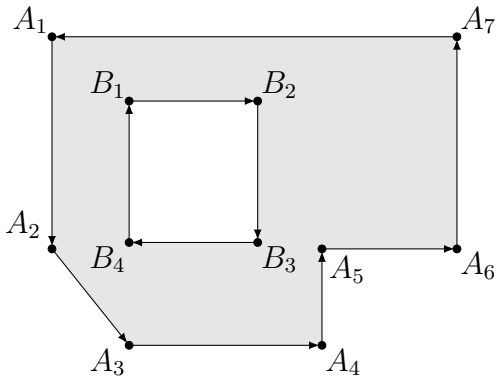


Figura 4.14: Exemplo de representação de uma região com furo.

correto. A inserção dos pontos de intersecção também não sofre alterações. As mesmas dificuldades existentes por ocasião da adoção da precisão finita se repetem para as arestas salientes. Os vértices salientes não são processados nesta etapa.

Para realizar as etapas seguintes, Leonov e Nikitin [40] adotaram uma estrutura chamada lista de conectividades (connective list) que possui descritores de todas as arestas que possuem vértices de cruzamento em uma intersecção especificada. Os descritores possuem o comprimento, o ângulo e a orientação de cada aresta. A lista de conectividades é montada durante o processo de detecção de intersecções. Cada vez que uma intersecção é inserida em uma aresta (saliente ou não), esta é adicionada à lista. Após todos os vértices de cruzamento serem determinados, a lista é ordenada em ordem crescente de ângulos das arestas. A Fig. 4.15 apresenta um exemplo de lista de conectividades. É importante notar que, se um mesmo polígono possuir dois vértices na mesma posição, estes possuirão índices distintos, inclusive no caso de vértices de arestas salientes. Sendo assim, cada vértice de contorno possui dois descritores D^+ e D^-, enquanto que um vértice proveniente de arestas saliente possui apenas um descritor D^s.

Na realização da classificação das arestas, a posição relativa (interna, externa ou coincidente) é determinada através da lista de conectividades. A primeira verificação que deve ser realizada é se a aresta coincide com outra. Neste caso só é necessário procurar por ângulos iguais em descritores de dois polígonos distintos e comparar sua orientação. Caso não seja encontrada uma aresta coincidente, deve-se analisar apenas os descritores do polígono que não possui a aresta. Busca-se pelo primeiro descritor anterior e o posterior ao da aresta que se deseja classificar. A aresta será rotulada como:

- INTERNA: caso o anterior seja um descritor D^- e o posterior D^+.
- EXTERNA: caso o anterior seja um descritor D^+ e o posterior D^-.

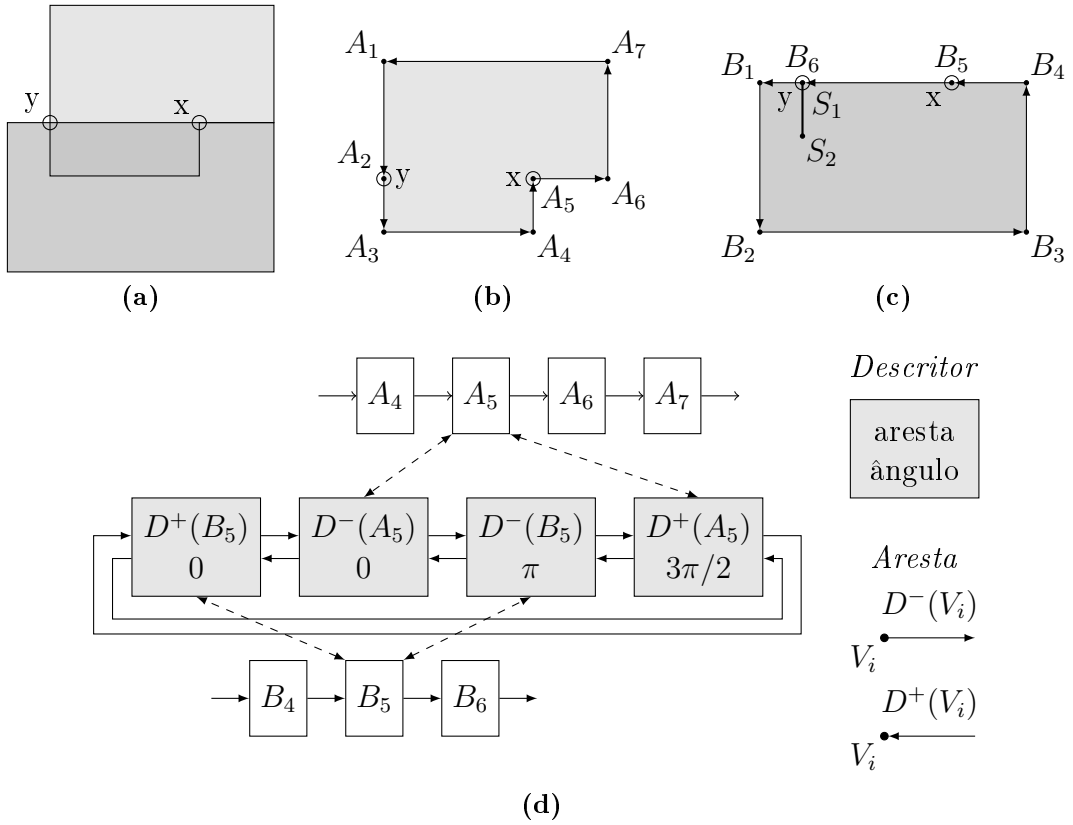


Figura 4.15: Exemplo de lista de conectividades. (a) Polígonos A e B, com a localização de x e y indicados. (b) Identificadores atribuídos aos vértices de A. (c) Identificadores atribuídos aos vértices de B, incluindo de arestas salientes S_1 e S_2. (d) Acima: lista de vértices do contorno de A. Abaixo: lista de vértices do contorno de B. Centro: lista de conectividades de x.

Os dois casos estão ilustrados na Fig. 4.16. Como deseja-se determinar as posições relativas das arestas salientes para que se possa aplicar os rótulos definidos, estas também devem constar na lista de conectividades. Entretanto, cuidados devem ser tomados para que esta inclusão não altere a classificação das arestas de contorno, uma vez que já foi visto que os resultados dos contornos devem ser os mesmos para a operação Booleana não regularizada. Os vértices salientes são classificados comparando sua posição em relação a cada um dos contornos, arestas e vértices salientes.

A Fig. 4.17 apresenta uma lista de conectividades que possui um descritor de aresta saliente. Para classificar a aresta $D^-(A_2)$ analisam-se os descritores $D^-(B_6)$ e $D^+(B_6)$, anterior e posterior respectivamente, desconsiderando o descritor da aresta saliente $D^s(S_1)$. Assim, a aresta é rotulada como INTERNA. No entanto, o descritor da aresta $D^s(S_1)$ deve estar presente na lista para se rotular a aresta saliente como COMPARTILHADO, uma vez que $D^-(A_2)$ possui o mesmo ângulo $3\pi/2$.

A coleta de arestas segue a ordem dos contornos de entrada. Quando um vértice de cruzamento é identificado, isto é, quando existe mais de uma possibilidade para a escolha da próxima aresta, a lista de conectividades é consultada para escolher a próxima aresta da lista que será incluída. Da mesma forma que na etapa de classificação, a aresta saliente não deve influenciar na escolha para definir o contorno. Deve-se constar também que, como a coleta de arestas e vértices salientes é realizada de forma individual, a ordem em que são coletadas não influencia o resultado. Assim, a lista de conectividades não é necessária para a coleta de elementos salientes.

A geração de arestas e vértices salientes é outra funcionalidade adicionada ao algoritmo original. Esta é realizada depois da coleta de arestas de contorno. Isto porque ela exige a verificação de todas as arestas, não somente das percorridas durante a coleta. Pode ser considerada como uma nova etapa, que necessita apenas que a classificação esteja finalizada.

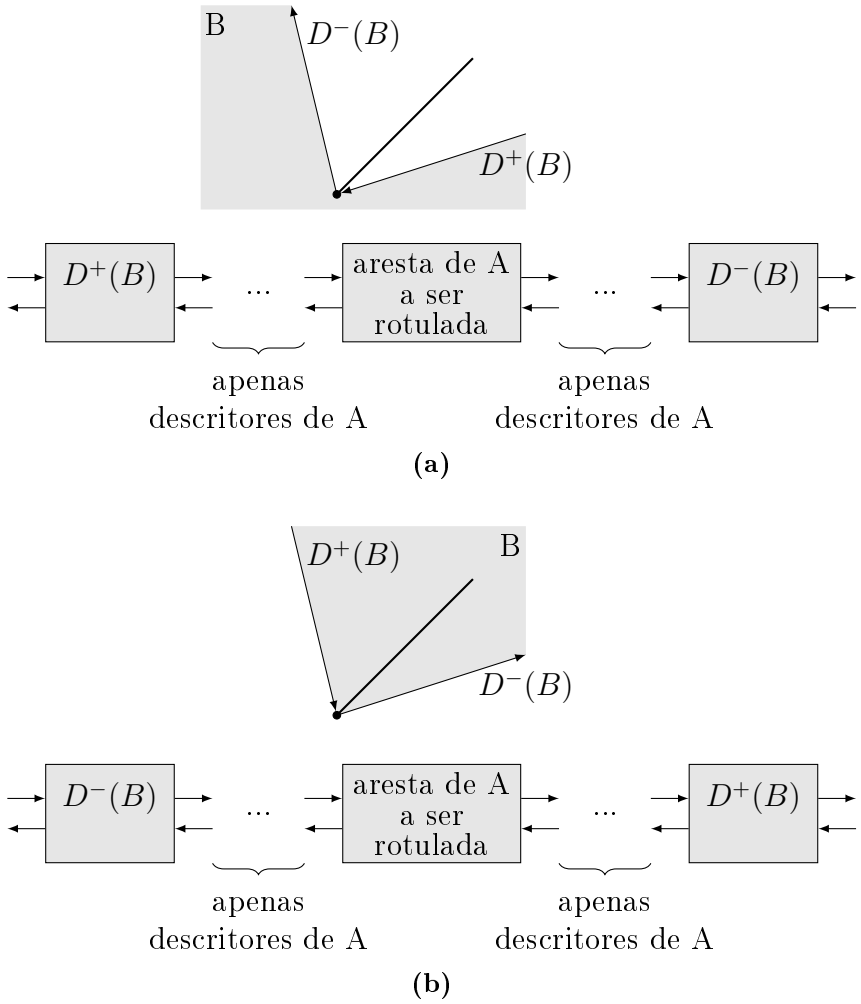


Figura 4.16: Classificação de arestas utilizando lista de conectividades. Os descritores são ordenados em ordem crescente de ângulo (da esquerda para direita) e a lista é circular. Considere uma operação A op B, onde op pode ser $-$ ou $\cup$. As arestas orientadas definem a região B, em cinza. A aresta que se deseja rotular é representada em destaque. Pode não existir nenhum descritor entre os laterais e o central na figura. (a) A aresta é externa a região B. (b) A aresta é interna a região B.

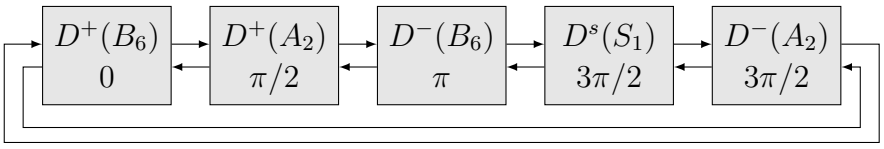


Figura 4.17: Lista de conectividades para o ponto y da Fig. 4.15.

5 Algoritmos propostos

Utilizando o conceito de região livre de colisão, é possível definir leiautes factíveis dada uma sequência de itens. Estes itens devem ser posicionados em algum ponto da região livre de colisão. Foi estabelecido que este posicionamento deve ser realizado apenas nos vértices da região livre de colisão. Como podem existir vértices ou arestas salientes, que representam posições de melhor compactação, estas regiões devem ser priorizadas no posicionamento de um novo item. Utilizando o algoritmo de recozimento simulado, foram definidos dois algoritmos para resolver os problemas dual e primal.

A seção 5.1 apresentação a decomposição adotada para o problema de empacotamento. O recozimento simulado é descrito na seção 5.3. Em seguida é apresentado o algoritmo para resolver o problema primal, com posicionamento rotacional, na seção 5.4. A seção 5.5 trata do problema dual, cuja solução é derivada do algoritmo anterior.

5.1 Decomposição do macro–problema de posicionamento

Como visto, a região livre de colisão fornece a informação geométrica sobre a região onde o posicionamento sem colisão é possível para um único item, dadas as posições dos itens já posicionadas no contêiner. Isso implica que o posicionamento dos itens deve ser feito de forma sequencial. Assim, em primeiro lugar, os itens devem ser ordenadas de acordo com algum critério. Em seguida, estes devem ser inseridos no contêiner de modo que cada item seja posicionado sem sobreposição desta com qualquer um dos itens já posicionados. Assim, o macro-problema de se determinar o posicionamento dos itens pode ser decomposto nos seguintes sub–problemas:

- Determinar uma ordem de posicionamento para os itens. Este sub–problema trata da escolha de quais itens serão selecionados para o posicionamento,

visto que os últimos itens a serem posicionadas encontram um contêiner já obstruído. Além disso, ele exerce uma forte influência na posição final dos itens selecionados. Diversos trabalhos relacionados ao problema dual (nos quais o problema de seleção de itens inexiste) tratam *exclusivamente* da determinação da ordem de posicionamento, aplicando em seguida uma heurística inferior esquerdo (*bottom–left*) à sequência dos itens (por exemplo, em [3] e em [43]).

- Determinar as rotações de cada item. Este sub–problema é o que caracteriza o posicionamento rotacional. Como a geometria dos itens foi considerada irrestrita, este é um problema combinatorial por si, e de difícil solução. Uma solução algorítmica para mínimos locais (com rotações limitadas) é apresentada em [44].

- Determinar para cada um dos itens a sua translação relativa aos demais e ao contêiner. Este sub–problema é o mais estudado na literatura. Como já mencionado, pretende-se empregar o polígono de obstrução para determinar as regiões livres de colisão, reduzindo o problema à determinação do ponto nestas regiões livres sobre o qual o item será posicionado.

Dois algoritmos foram propostos, um para resolver o problema dual, em que o contêiner possui uma dimensão variável a ser minimizada, e outro para o prolema primal, em que o contêiner possui dimensões fixas e se deseja encaixar todos os itens.

5.2 Heurística de posicionamento adotada

No algoritmo TOPOS [17], o posicionamento de um novo item ocorre apenas nos vértices do polígono de obstrução ou em pontos obtidos por uma busca estendida (extended search). Esta heurística foi baseada no trabalho de Adamowicz e Albano [45], em que foi adotado o critério de enquadramento dos itens para avaliar as soluções parciais, isto é, soluções em que existem itens não posicionados. Adotando este critério, o posicionamento nos vértices ou nos pontos obtidos por uma busca estendida é um mínimos local [45]. Os resultados obtidos por [17] e por [16], que utiliza um algoritmo TOPOS adaptado, mostram a eficiência desta heurística de posicionamento.

O problema considerado pelo algoritmo TOPOS [17] é a variante dual. No caso do problema primal, os critérios utilizados em [17] não se adaptam muito

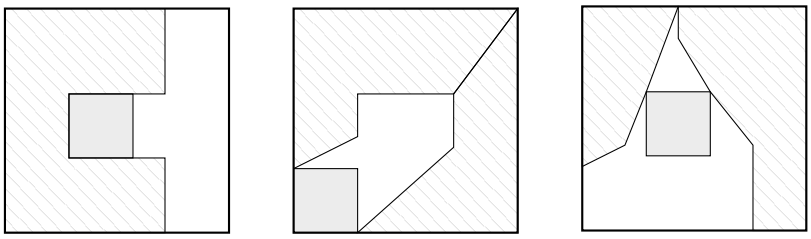


Figura 5.1: Exemplos de posição de encaixe para o item móvel em cinza. Itens fixos estão representados por polígonos hachurados.

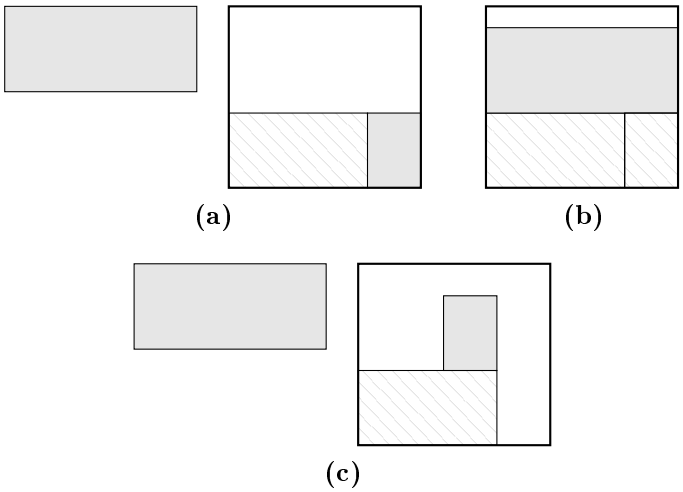


Figura 5.2: (a) e (b) demonstram o posicionamento do item móvel em cinza em uma aresta saliente (posição de encaixe). Itens fixos estão representados por polígonos hachurados. Isso possibilita o posicionamento de um terceiro item. (c) O não posicionamento de encaixe não possibilita a inserção de um novo item.

bem. Por este motivo, uma nova heurística de posicionamento será adotada. No problema primal, posições de encaixe são muito importantes. Posições de encaixe são posições em que o item móvel encosta em algum item fixo ou contêiner em pelo menos dois pontos (vide Fig. 5.1). O posicionamento de uma peça em uma posição de encaixe muitas vezes possibilita o posicionamento de mais peças no contêiner, como no exemplo da Fig. 5.2. É fácil constatar que os problemas do tipo quebra–cabeça só podem ser resolvidos com o posicionamento de pelo menos um item encaixado. Utilizando o conceito de região livre de colisão, calculada a partir de operações Booleanas não regularizadas, algumas posições de encaixe são facilmente identificadas: arestas e vértices salientes. Desta forma, no algoritmo desenvolvido, estas posições foram priorizadas.

Outra posição de encaixe são os vértices côncavos do polígono de obstrução que considera todos os itens fixos, ou seja, a união de todos os polígonos de obstrução. O posicionamento de um item convexo em um vértice côncavo do polígono de obstrução ocasiona o encaixe do item na região resultante da união dos itens fixos (Fig. 5.3a). Já no caso de um item móvel não convexo, o posicionamento em

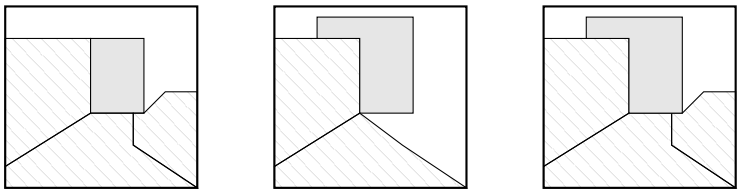


Figura 5.3: Itens fixos estão representados por polígonos hachurados e o item móvel por um polígono preenchido de cinza. (a) Encaixe de um item convexo. (b) Encaixe de um item não convexo. (c) Encaixe de um item não convexo em um leiaute não convexo.

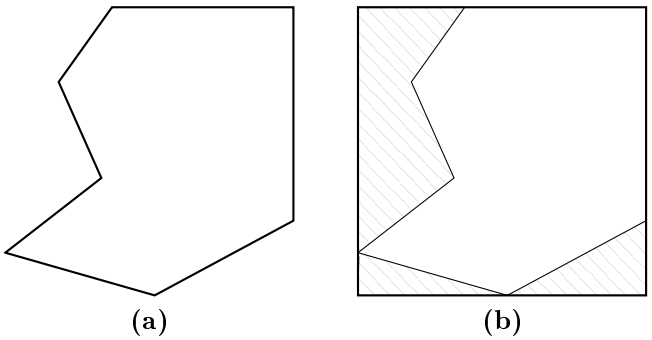


Figura 5.4: (a) Contêiner irregular (b) Contêiner retangular com obstáculos (polígonos hachurados), que possui a mesma região interna livre que o contêiner irregular.

um vértice côncavo do polígono de obstrução representa o encaixe da região resultante da união dos itens fixos no item móvel (Fig. 5.3b). Também pode ocorrer uma combinação dos dois casos, em que ambas as regiões são não convexas, como no exemplo da Fig. 5.3c. Como o cálculo da região livre de colisão não assume que a união de todos os polígonos de obstrução foi realizada, não é possível detectar todos estes pontos de encaixe. No entanto, considerando contêiner retangular, vértices côncavos da região livre de colisão só são originados de vértices convexos da região resultante da união de todos os polígonos de obstrução. Como estes vértices não são posições de encaixe, devem ter menor prioridade no posicionamento. Para o caso de contêineres não retangulares, pode-se considerá-los como uma composição de um contêiner retangular e obstáculos (exemplo mostrado na Fig. 5.4). Desta forma, o mesmo procedimento de escolha de vértices de contornos para o posicionamento pode ser realizado.

A heurística de posicionamento adotada nos algoritmos desenvolvidos possui a seguinte ordem de prioridade: posicionamento em um vértice saliente, vértice de uma aresta saliente e vértices convexos de contorno. Se a região livre de colisão possuir mais que um vértice saliente, a heurística escolhe de forma aleatória um deles. Se não possuir vértices salientes, mas sim arestas salientes, um sorteio será realizado para escolher um dos vértices dessas arestas. Finalmente, caso existam

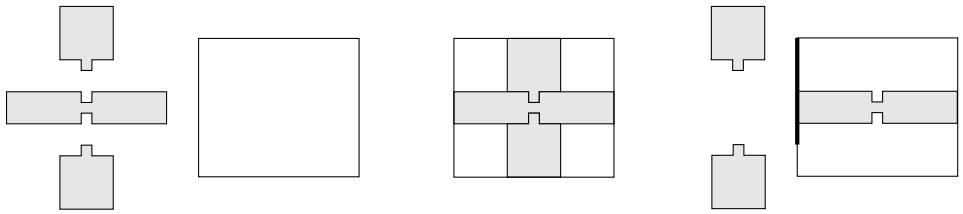


Figura 5.5: (a) Este problema de posicionamento possui três itens na esquerda e um contêiner retangular na direita. (b) Único leiaute possível para o encaixe das três peças sem sobreposição. (c) Considerando que o primeiro item a ser posicionado é o item central, e seu ponto de referência coincide com o vértice superior mais à esquerda, a sua região livre de colisão é um segmento de reta representado pela linha em destaque e deve ser posicionado em um ponto intermediário deste segmento. O segundo e terceiro itens devem ser posicionados em um vértice da sua região livre de colisão. Não importa a ordem de posicionamento das peças, o primeiro item nunca deve ser posicionado no vértice de sua região livre de colisão para se obter a solução correta.

apenas contornos fechados, a heurística de posicionamento escolhe um dos vértices convexos dos contornos. Existem casos em que a utilização desta heurística não converge para o melhor resultado, no entanto, são casos muito especiais como o da Fig. 5.5.

5.3 Recozimento simulado

O recozimento simulado [19] é a meta-heurística probabilística adotada neste trabalho. A sua escolha se deve a capacidade de escapar de mínimos locais, que são frequentes nos problemas estudados.

O recozimento simulado se originou a partir do algoritmo de Metropolis [46], que é capaz de simular a configuração de um grupo de átomos em equilíbrio a uma dada temperatura. Kirkpatrick et al. [19] observou a semelhança entre o comportamento de um sistema de vários graus de liberdade em equilíbrio térmico e o problema combinatorial multivariável. Assim, propôs aplicar o algoritmo de Metropolis para simular a recristalização dos átomos durante o processo de recozimento (esfriamento gradual e controlado). Durante o recozimento, os átomos migram naturalmente para a configuração que minimiza a energia do sistema, mesmo que durante o processo o sistema passe por configurações de maior energia.

Durante a execução do algoritmo do recozimento simulado, modificações aleatórias na solução são aplicadas e a função objetivo é recalculada. Se o valor da função for menor do que a anterior, esta é aceita. Caso contrário, a solução deve

ser aceita com a probabilidade dada pela distribuição de Boltzman:

$$P(\Delta E) = e^{-\frac{\Delta E}{kt}} \tag{5.1}$$

onde ΔE é a diferença entre o valor atual da função objetivo e a anterior, $P(\Delta E)$ é a probabilidade de se aceitar uma solução que acarreta em uma maior função objetivo, k é um parâmetro do processo (análogo a constante de Stefan-Boltzman) e t é a temperatura atual. O processo de decréscimo desta temperatura deve ser controlado, e é o principal parâmetro do algoritmo.

5.4 Problema primal

O problema primal estudado é o de posicionar todos os itens dentro de um contêiner com dimensões fixas. O contêiner pode ser um polígono irregular qualquer, não necessariamente convexo. Cada item pode ser rotacionado de qualquer ângulo. Assim, no algoritmo proposto, o recozimento simulado controla a sequência de posicionamento, o ângulo de rotação e a posição de cada item. A função objetivo é o espaço não ocupado no contêiner. Esta função assume valores discretos, ao passo que o ângulo de rotação de cada peça é considerado contínuo.

Como visto na Fig. 2.2, a função objetivo possui apenas valores discretos e apresenta regiões de valor constante. Isto decorre do fato de que o valor da função objetivo depende apenas da identificação de quais itens foram posicionados e quais não foram. Como podem existir diversas soluções em que o mesmo conjunto de itens é posicionado, é razoável considerar que as soluções em que o posicionamento de um novo item está mais próximo de ocorrer são melhores. Esta proximidade pode ser avaliada através da tentativa de se posicionar uma versão de dimensões reduzidas de algum dos itens restantes. Assim, para cada item não encaixado, é executada uma busca binária com profundidade limitada para encontrar um fator de escala (entre 0 e 1) que, aplicado ao item, possibilita o seu encaixe [9]. Se o encaixe for bem sucedido, a área deste item é removida da função objetivo.

O algoritmo ResolvePrimal corresponde ao recozimento simulado proposto em [24] e é uma abordagem genérica para resolver o problema de posicionamento através do controle simultâneo de três tipos de parâmetros: sequência de posicionamento, rotação e translação. A condição global de parada, ou de convergência, se dá quando, na temperatura dada do algoritmo de recozimento simulado, forem aceitas somente soluções equivalentes à melhor encontrada. A condição local é um número máximo de iterações pré-definido. Alguns dos controles de parâ-

metro (sequência, rotação ou translação do item) podem ser desabilitados e o recozimento simulado pode ser combinado com heurísticas determinísticas.

Algoritmo 1: ResolvePrimal.

```
x ← <Solução inicial aleatória>
T_0 ← <Temperatura inicial>
enquanto <Condição global de parada não satisfeita> faça
    T_i ← T_i * α; i ← i + 1
    enquanto <Condição local de parada não satisfeita> faça
        val ← random(0, 1)
        se val < 1/3 então
            x* ←<Modificar sequência de posicionamento>
        senão
            <Seleciona o item móvel>
            se val < 2/3 então
                x* ←<Aplica nova rotação ao item>
            senão
                x* ←<Seleciona um vértice da RLC para posicionar>
        se <possuir itens não posicionados> então
            <tenta posicionar item escalado>
        ΔE = F(x*) − F(x)
        se ΔE < 0 então
            x ← x*
        senão
            se random(0, 1) < e^(−ΔE/kT) então x ← x*
```

5.4.1 Heurísticas Determinísticas

Diversos pesquisadores combinam as heurísticas probabilísticas com determinísticas [6, 12]. As determinísticas são geralmente utilizadas para definir a ordem de posicionamento e a translação do item. As heurísticas conhecidas como maior primeiro (larger first), que são as heurísticas de ordenação mais estudadas [22], organizam os itens em ordem decrescente de tamanho. São baseadas na observação de especialistas que lidam com problemas de empacotamento, que tendem a posicionar primeiro os itens maiores no contêiner. Como é comum em heurísticas determinísticas, é fácil criar problemas em que estas não convergem para o ótimo global. A Fig. 5.6 apresenta um problema que não pode ser solucionado com a combinação das heurísticas maior primeiro e a heurística construtiva adotada neste trabalho, em que novos itens são posicionados de forma que encostem em itens já posicionados ou no contêiner, ou seja, nos vértices da região livre de colisão.

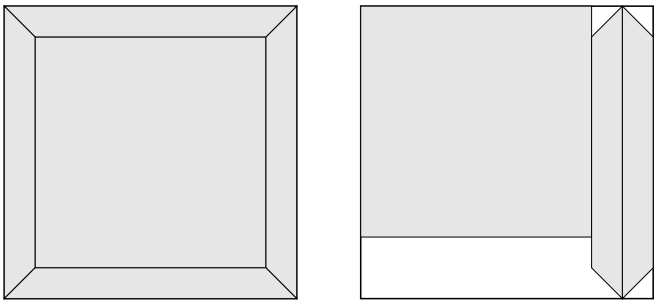


Figura 5.6: Quebra-cabeça que não pode ser resolvido com a heurística maior primeiro. (a) Solução para o problema. (b) Posicionamento utilizando maior primeiro. Como o posicionamento deve ocorrer nos vértices da região livre de colisão, o item retangular não pode ser posicionado no centro. Assim, não é possível resolver o quebra-cabeça.

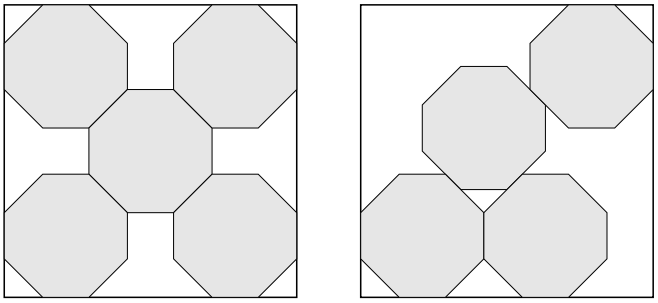


Figura 5.7: Quebra-cabeça que não pode ser resolvido com a heurística inferior esquerdo. (a) Solução para o problema. (b) Posicionamento utilizando inferior esquerdo. O primeiro item é posicionado corretamente, no entanto, como o item seguinte deve ser inserido na posição mais inferior e encostado no primeiro, não será possível solucionar o problema.

A heurística determinística mais comum para o posicionamento de um item é a inferior esquerdo. Nesta abordagem, os itens são posicionados no contêiner na posição livre mais inferior e mais à esquerda disponível. A popularidade da heurística inferior esquerdo pode ser compreendida pelo seu baixo custo computacional. Além disso, ao agrupar os itens perto das paredes do contêiner mantém-se uma grande área desobstruída. Deve-se notar também que é fácil produzir um problema de empacotamento em que o ótimo global não pode ser alcançado pela heurística inferior esquerdo (veja Fig. 5.7).

5.5 Problema dual

É considerado o problema dual em que o contêiner é retangular e possui uma dimensão fixa e outra variável. O objetivo é obter um leiaute factível, com todos os itens posicionados, utilizando um contêiner com a menor dimensão variável.

Os itens podem ser rotacionados de 90°, 180°, e 270°, possuindo até quatro orientações possíveis.

O algoritmo ResolveDual proposto para o problema dual possui dois níveis hierárquicos. O nível interno é o mesmo algoritmo utilizado para resolver o problema primal, desconsiderando-se as rotações.

O nível externo controla o recozimento simulado definindo a temperatura e leiaute iniciais, assim como o valor da dimensão variável. Para este nível são definidos dois novos parâmetros p_{dim} e p_{aum}, que assumem valores entre 0 e 1. Uma vez que o nível interno converge para um leiaute factível em que todos os itens foram posicionados, este nível é abandonado e o nível externo multiplica o comprimento da dimensão variável por $(1 - p_{dim})$, diminuindo o contêiner, e reinicia a execução do recozimento simulado. Caso o recozimento simulado convirja para uma solução que não possui todos os itens posicionados, o nível externo aumenta o comprimento da dimensão variável, multiplicando por $(1 + p_{aum})$, aumenta temperatura e executa o recozimento simulado mais uma vez.

Algoritmo 2: ResolveDual.

```
x ← <Solução inicial aleatória>
T_0 ← <Temperatura inicial>
L ← <Comprimento inicial do contêiner>
enquanto <Não terminado> faça
    enquanto <Condição global de parada não satisfeita> faça
        T_i ← T_i * α; i ← i + 1
        enquanto <Condição local de parada não satisfeita> faça
            val ← random(0, 1)
            se val < 0.5 então
                x* ←<Modificar sequência de posicionamento>
            senão
                <Seleciona o item móvel>
                x* ←<Seleciona um vértice da RLC para posicionar>
            ΔE = F(x*) − F(x)
            se ΔE < 0 então
                x ← x*
            senão
                se random(0, 1) < e^{−ΔE/kT} então x ← x*
            se <Todos os itens foram posicionados> então
                L ← (1 − p_dim)L
                <Comprimento do contêiner>← L
                <Condição global de parada satisfeita>
    se <Algum item não foi posicionado> então
        L ← (1 + p_aum)L
        <Comprimento do contêiner>← L
        i ← 0; T_0 ← <Temperatura inicial>
```

6 Resultados

Neste capítulo são apresentados os resultados obtidos para os dois problemas estudados: primal e dual. Também foi realizada a medição dos tempos obtidos nos testes de determinação da região livre de colisão, utilizando o algoritmo paralelo e o serial.

Para testar o algoritmo primal (na seção 6.1), foram adotados problemas que possuem ótimo global conhecido. Variações com furos e contêineres convexos também foram resolvidos. Por fim, foi realizada a comparação com resultados obtidos em trabalhos anteriores. Na seção 6.2 o algoritmo dual é avaliado utilizando problemas encontrados na literatura. São obtidos alguns leiautes mais compactos do que os encontrados na literatura. Na última seção, 6.3, para cada caso da literatura é adotado um conjunto de polígonos de obstrução e um polígono de posicionamento interno. Estes constituem os dados utilizados para se medir o tempo de execução para a determinação da região livre de colisão utilizando os métodos propostos.

6.1 Resultados do problema primal

Os problemas estudados, adotados de [47], possuem solução em que todos os itens podem ser posicionados no interior do contêiner. Assim, os problemas possuem solução ótima global conhecida, o que é desejável para avaliar os testes. Ao analisar os resultados, é importante ressaltar que uma solução tão compacta como a final pode ser encontrada em um número menor de iterações. No entanto, o único modo de garantir que a solução compacta encontrada é a melhor é esperar até o algoritmo convergir. Mesmo que o algoritmo seja interrompido durante sua execução, é possível obter uma solução satisfatória.

Foram testadas sete heurísticas:

- Completa, em que o posicionamento dos itens é rotacional e realizado nos vértices da região livre de colisão e a sequência de posicionamento dos itens

é controlada pelo recozimento simulado;

- inferior esquerdo, em que o posicionamento dos itens é rotacional e obedece a heurística inferior esquerdo e a sequência de posicionamento dos itens é controlada pelo recozimento simulado;
- maior primeiro, em que o posicionamento dos itens é rotacional e realizado nos vértices da região livre de colisão e a sequência de posicionamento dos itens obedece a heurística maior primeiro;
- maior primeiro e inferior esquerdo, em que o posicionamento dos itens é rotacional e obedece a heurística inferior esquerdo e a sequência de posicionamento dos itens obedece a heurística maior primeiro, neste caso o único parâmetro controlado pelo recozimento simulado é a rotação de cada item;
- translacional, em que o posicionamento dos itens é translacional e realizado nos vértices da região livre de colisão e a sequência de posicionamento dos itens é controlada pelo recozimento simulado;
- translacional inferior esquerdo, em que o posicionamento dos itens é translacional e obedece a heurística inferior esquerdo e a sequência de posicionamento dos itens é controlada pelo recozimento simulado;
- translacional maior primeiro, em que o posicionamento dos itens é translacional e realizado nos vértices da região livre de colisão e a sequência de posicionamento dos itens obedece a heurística maior primeiro.

Os casos em que a heurística aplicada não possibilita a convergência do problema para a solução ótima não foram testados.

Todos os testes foram executados em um computador que possui um processador Phenon 9550 2.21Ghz. As Tabelas 6.1–6.6 apresentam os resultados para solução dos problemas. Em cada uma são apresentadas colunas com as médias para 30 execuções, a primeira coluna apresenta a heurística adotada, a segunda coluna a profundidade máxima utilizada na busca binária, N_{conv} representa o número de avaliações da função objetivo até atingir a convergência, N_{min} é o número mínimo de avaliações da função objetivo para se determinar o seu menor valor observado, T_{conv} é o tempo de execução em segundos até atingir a convergência e P_{conv} representa o número de execuções que atingiu o mínimo global em porcentagem.

6.1.1 Tangram

O tangram é um quebra-cabeça que consiste de sete itens congruentes convexos. A Fig. 6.1 mostra configurações finais possíveis para este problema. A Tabela 6.1 apresenta os resultados para solução deste problema.

No posicionamento translacional, todas as execuções convergiram para mínimo global (sem desperdício de espaço no contêiner). O posicionamento rotacional necessitou de 764 vezes mais iterações e seu tempo de execução foi 4432 vezes maior quando comparado com o problema translacional.

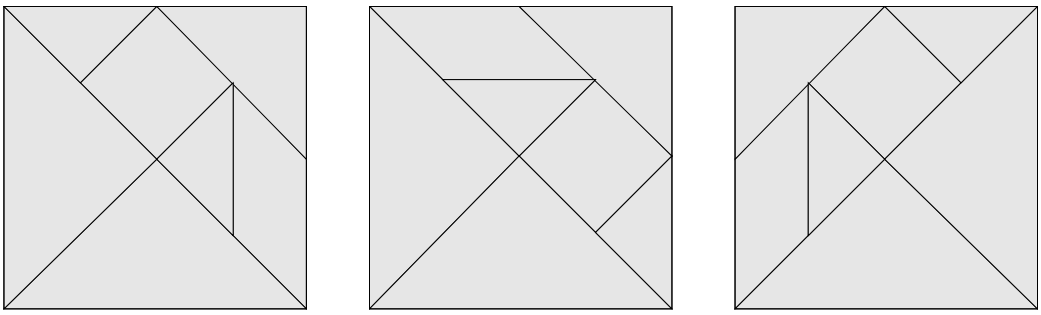


Figura 6.1: Soluções ótimas de um tangram com sete itens.

6.1.2 Contêiner com furo

O problema do contêiner com furo é uma variante do tangram. Para representar o furo, são inseridos itens que ocupam a região e permanecem fixos durante a execução do algoritmo. Como este furo é não–convexo, é necessário realizar a decomposição convexa manualmente em uma etapa de pré–processamento. A Fig. 6.2 apresenta soluções finais possíveis para o problema de contêiner com furo. A Tabela 6.2 apresenta os resultados para solução deste problema.

Todos os testes, exceto o que não utiliza heurísticas e o que utiliza heurística inferior esquerdo, ambos sem realizar a busca binária, convergiram para o melhor resultado possível. Os testes translacionais atingiram o ótimo global em 213 vezes menos iterações e com um tempo de execução 828 vezes menor do que os equivalentes rotacionais.

6.1.3 Quebra–cabeça Simples

Este problema é um quebra–cabeça relativamente simples com quatro itens não–convexos. Os itens não–convexos são decompostos em uma etapa de pré-processamento, sendo que esta decomposição não afeta o resultado final. A Fig. 6.3 mostra soluções finais para este problema.

Tabela 6.1: Estatísticas para o tangram. Foi adotado $\alpha = 0.98$ para os casos rotacionais e $\alpha = 0.55$ para os translacionais. Neste problema não é possível atingir o ótimo global utilizando as heurísticas inferior esquerdo e maior primeiro simultaneamente

Heurística(s) adotada(s)	Profundidade	N_{conv}	N_{min}	T_{conv}	P_{conv}
Nenhuma	0	130169	57724	$722,83$	$58,8\%$
	1	183939	100608	$2209,01$	$90,9\%$
	2	225041	84950	$3937,47$	$82,4\%$
Inferior esquerdo	0	101935	49696	$643,30$	$77,4\%$
	1	163613	78469	$2413,91$	$90,3\%$
	2	186065	68163	$3013,95$	$83,9\%$
Maior primeiro	0	99809	32276	$360,47$	$90,3\%$
	1	140251	57910	$1705,05$	$90,3\%$
	2	155617	47279	$2063,22$	$86,7\%$
Translacional	0	261	98	$0,30$	$100,0\%$
	1	275	124	$0,57$	$100,0\%$
	2	282	104	$0,82$	$100,0\%$
Translacional inferior esquerdo	0	206	38	$0,31$	$100,0\%$
	1	228	45	$0,56$	$100,0\%$
	2	209	37	$0,65$	$100,0\%$
Translacional maior primeiro	0	133	25	$0,12$	$100,0\%$
	1	135	28	$0,21$	$100,0\%$
	2	152	39	$0,31$	$100,0\%$

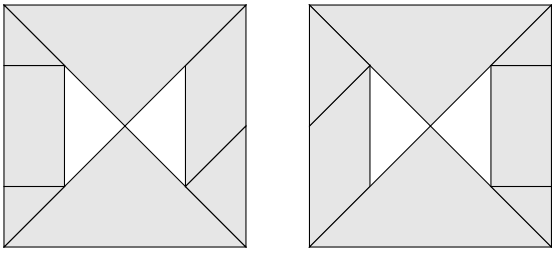


Figura 6.2: Soluções ótimas do contêiner com furo.

Tabela 6.2: Estatísticas para o contêiner com furo. Foi adotado $\alpha = 0.98$ para os casos rotacionais e $\alpha = 0.55$ para os translacionais

Heurística(s) adotada(s)	Profundidade	N_{conv}	N_{min}	T_{conv}	P_{conv}
Nenhuma	0	12847	10617	28,89	93,5%
	1	15057	12651	206,51	100,0%
	2	13990	11295	254,61	100,0%
Inferior esquerdo	0	10025	7992	30,67	87,5%
	1	11470	9161	156,51	100,0%
	2	11963	8431	250,85	100,0%
Maior primeiro	0	7837	4413	17,61	100,0%
	1	7443	3955	50,25	100,0%
	2	9674	3804	120,64	100,0%
Inferior esquerdo e Maior Primeiro	0	8160	2808	50,91	100,0%
	1	9181	2569	130,66	100,0%
	2	15117	2417	419,65	100,0%
Translacional	0	75	42	0,10	100,0%
	1	89	58	0,22	100,0%
	2	78	47	0,27	100,0%
Translacional inferior esquerdo	0	52	25	0,09	100,0%
	1	51	22	0,17	100,0%
	2	53	25	0,23	100,0%
Translacional maior primeiro	0	33	7	0,03	100,0%
	1	27	2	0,05	100,0%
	2	39	12	0,09	100,0%

A partir da Tabela 6.3, que apresenta os resultados para solução deste problema, constata-se que o tempo de execução para as variantes translacionais convergiram em 57 vezes menos iterações e com tempo de execução 340 vezes menor do que as rotacionais. Todos os testes convergiram para o leiaute com contêiner plenamente ocupado.

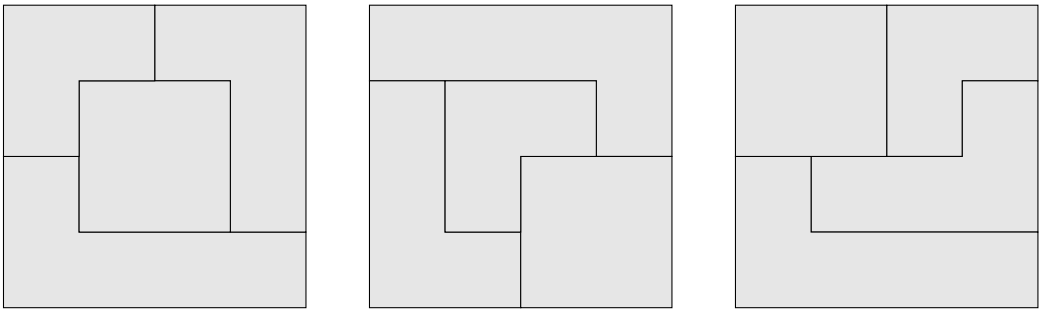


Figura 6.3: Soluções finais para o quebra–cabeça simples com quatro itens.

6.1.4 Contêiner Côncavo

O problema da Fig. 6.4 ilustra a capacidade do processo de tratar problemas com contêineres não convexos. O contêiner não convexo é representado por um contêiner convexo com dois furos. A Tabela 6.4 apresenta os resultados para solução deste problema.

Os dados da tabela mostram a variação exponencial do tempo de execução quando analisada a influência da busca binária. O posicionamento translacional ocasionou um tempo de execução 1210 vezes menor e foi capaz de atingir o ótimo global em 193 vezes menos iterações do que o posicionamento rotacional.

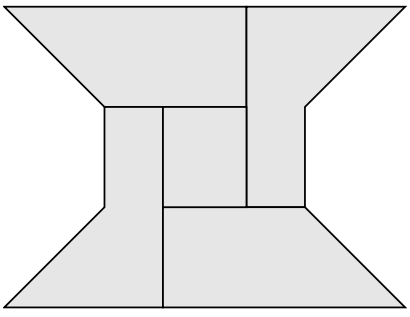


Figura 6.4: Solução final para o problema de contêiner côncavo.

6.1.5 Maior primeiro falha

O problema maior primeiro falha consiste no posicionamento de cinco itens convexos. Este problema não converge para o ótimo global se for utilizada a heurística maior primeiro. A Fig. 6.5 mostra a solução final para este problema.

Tabela 6.3: Estatísticas para o quebra–cabeça simples. Foi adotado $\alpha = 0.98$ para os casos rotacionais e $\alpha = 0.55$ para os translacionais.

Heurística(s) adotada(s)	Profundidade	N_{conv}	N_{min}	T_{conv}	P_{conv}
Nenhuma	0	24604	17727	49,58	100,0%
	1	25364	19327	169,17	100,0%
	2	23068	16783	316,68	100,0%
Inferior esquerdo	0	22796	14455	52,78	100,0%
	1	22168	14494	152,57	100,0%
	2	27852	15698	378,05	100,0%
Maior primeiro	0	23521	14262	41,56	100,0%
	1	24542	11685	161,56	100,0%
	2	29584	14717	340,39	100,0%
Inferior esquerdo e Maior Primeiro	0	20748	7383	63,61	100,0%
	1	24311	10807	209,40	100,0%
	2	25941	9456	399,27	100,0%
Translacional	0	276	27	0,16	100,0%
	1	275	38	0,44	100,0%
	2	290	33	0,72	100,0%
Translacional inferior esquerdo	0	673	9	0,41	100,0%
	1	685	11	1,10	100,0%
	2	654	14	0,40	100,0%
Translacional maior primeiro	0	545	19	0,23	100,0%
	1	527	15	0,66	100,0%
	2	553	16	1,17	100,0%

Tabela 6.4: Estatísticas para o contêiner côncavo. Foi adotado $\alpha = 0.98$ para os casos rotacionais e $\alpha = 0.55$ para os translacionais. Neste problema não é possível atingir o ótimo global utilizando as heurísticas inferior esquerdo e maior primeiro simultaneamente

Heurística(s) adotada(s)	Profundidade	N_{conv}	N_{min}	T_{conv}	P_{conv}
Nenhuma	0	17912	14642	43,96	86,7%
	1	20923	19397	201,05	100,0%
	2	16032	13845	260,99	100,0%
Inferior esquerdo	0	26305	21339	72,39	83,3%
	1	19808	18026	236,40	100,0%
	2	15236	11335	261,95	96,8%
Maior primeiro	0	12945	8025	21,77	64,5%
	1	18933	17211	215,49	100,0%
	2	20341	15566	332,89	96,8%
Translacional	0	95	37	0,09	100,0%
	1	90	32	0,24	100,0%
	2	91	40	0,42	100,0%
Translacional inferior esquerdo	0	161	12	0,17	76,7%
	1	122	14	0,41	86,7%
	2	150	18	0,90	80,0%
Translacional maior primeiro	0	75	29	0,05	100,0%
	1	75	30	0,21	100,0%
	2	70	23	0,31	100,0%

A Tabela 6.5 mostra que o posicionamento com a heurística translacional atingiu a convergência com tempo de execução 300 vezes menor e número de iterações 116 vezes menor se comparado com os resultados obtidos com a heurística rotacional.

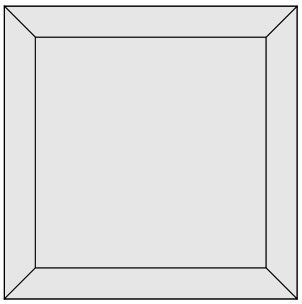


Figura 6.5: Solução final para o problema maior primeiro falha.

Tabela 6.5: Estatísticas para o maior primeiro falha. Foi adotado $\alpha = 0.98$ para os casos rotacionais e $\alpha = 0.55$ para os translacionais

Heurística(s) adotada(s)	Profundidade	N_{conv}	N_{min}	T_{conv}	P_{conv}
Nenhuma	0	4426	2818	$10,61$	$63,6\%$
	1	6276	5666	$26,35$	$100,0\%$
	2	5136	2901	$37,92$	$58,8\%$
Inferior esquerdo	0	4183	2884	$5,92$	$77,4\%$
	1	4750	4162	$20,14$	$100,0\%$
	2	4353	3241	$28,28$	$87,1\%$
Translacional	0	60	20	$0,05$	$100,0\%$
	1	39	12	$0,07$	$100,0\%$
	2	34	17	$0,08$	$100,0\%$
Translacional inferior esquerdo	0	69	9	$0,06$	$100,0\%$
	1	39	9	$0,07$	$100,0\%$
	2	34	6	$0,08$	$100,0\%$

6.1.6 Inferior esquerdo falha

Este problema não converge para o ótimo global se for utilizada a heurística inferior esquerdo. A Fig. 6.6 mostra a solução final para este problema. A Tabela 6.6 apresenta os resultados para solução deste problema. O tempo de execução aumentou em relação à profundidade da busca binária. O mínimo global foi atingido em todas as execuções.

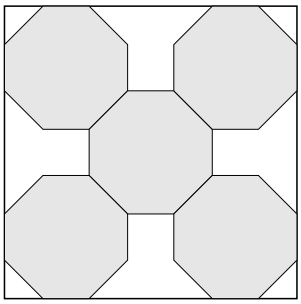


Figura 6.6: Solução final para o problema inferior esquerdo falha.

Tabela 6.6: Estatísticas para o problema translacional inferior esquerdo falha. Foi adotado $\alpha = 0.55$

Heurística(s) adotada(s)	Profundidade	N_{conv}	N_{min}	T_{conv}	P_{conv}
Translacional	0	117	14	0,16	100,0%
	1	116	11	0,28	100,0%
	2	120	15	0,35	100,0%
Translacional maior primeiro	0	112	11	0,12	100,0%
	1	123	11	0,21	100,0%
	2	117	13	0,29	100,0%

6.1.7 Comparação com trabalhos anteriores

A Tabela 6.7 apresenta uma comparação entre os resultados obtidos neste trabalho e alguns trabalhos publicados anteriormente. Os valores representam a razão entre os resultados dos trabalhos anteriores e os atuais.

Em [9], a heurística de posicionamento adotada definia que o item móvel deve ser posicionamento sobre vértices da região livre de colisão com 40% de probabilidade e sobre as arestas da região livre de colisão com 60% de probabilidade. Em [48], a heurística de posicionamento adotada definia que o item móvel deve ser posicionamento exclusivamente sobre vértices da região livre de colisão. Em ambas as propostas, a região livre de colisão foi calculada de modo aproximado por meio de operações Booleanas regularizadas.

Pode-se observar que o desempenho em todos os problemas melhoraram e a convergência foi atingida em menor tempo e em menor número de iterações. É importante notar que o tamanho do contêiner em trabalhos anteriores era expandido em aproximadamente 10.0% da área total dos itens. Neste trabalho, a área do contêiner é igual a soma das áreas dos itens. Desta forma, os itens se encaixam perfeitamente no contêiner, constituindo um problema de solução mais difícil, especialmente nos casos rotacionais.

Tabela 6.7: Comparação com resultados obtidos em trabalhos anteriores [9, 48]. Para cada trabalho, foi obtida a relação do número de iterações e tempo de execução entre os resultados dos trabalho citados e os deste. **CF**: contêiner com furo. **CC**: contêiner convexo. **QCS**: quebra–cabeça simples. **MP falha**: maior primeiro falha. **IE falha**: inferior esquerdo falha

	2009[48]				2010[9]			
	Rotacional		Translacional		Rotacional		Translacional	
Problema	N_{conv}	T_{conv}	N_{conv}	T_{conv}	N_{conv}	T_{conv}	N_{conv}	T_{conv}
Tangram	10,05	1,90	17,47	8,16	11,16	2,07	409,87	216,74
CF	4,09	0,98	20,55	22,55	67,62	19,66	35,31	41,22
CC	15,72	3,27	7,73	3,41	19,07	3,46	8,90	4,23
QCS	13,12	3,82	16,11	5,44	15,44	4,22	522,85	189,89
MP falha	11,95	3,08	14,90	5,68	11,87	2,99	19,85	7,84
IE falha			13,09	5,08			394,82	196,73

6.2 Resultados do problema dual

Neste trabalho, o algoritmo dual proposto foi avaliado utilizando 13 problemas da literatura, que podem ser encontrados no site da ESICUP (EURO Special Interest Group on Cutting and Packing). São problemas de itens irregulares com o objetivo de minimizar o comprimento de um contêiner retangular de largura fixa. O algoritmo desenvolvido busca por um contêiner de dimensão fixa de menor comprimento que contenha todos os itens sem sobreposição. Cada item pode rotacionar de um conjunto finito de ângulos (0°, 90°, 180° ou 270°), admitindo até quatro orientações possíveis. A Tabela 6.8 mostra os problemas da literatura e o número total de polígonos, número médio de vértices e orientações possíveis para cada caso. Todos os testes foram executados com um processador i7 860 com 4 GB de memória RAM.

Foram adotadas duas estratégias para resolver o problema dual. A primeira consiste em definir manualmente as dimensões do contêiner, inclusive a variável, e utilizar o algoritmo desenvolvido para o caso primal para realizar o posicionamento de todas os itens neste contêiner. Neste caso, o controle rotacional do recozimento simulado e a busca binária são desligados. Se o algoritmo convergir para um leiaute em que todas as peças são posicionadas no contêiner, o usuário deve diminuir o seu comprimento, caso contrário deve aumentá-lo. Os comprimentos mínimos e densidades obtidas por esta estratégia estão apresentados na Tabela 6.9.

A segunda estratégia é o algoritmo proposto na seção 5.5. Nesta, o nível externo do algoritmo é responsável por controlar a dimensão variável do contêi-

Tabela 6.8: Dados dos problemas encontrados na literatura. **NTP**: número total de polígonos; **NMV**: numero médio de vértices; **OP**: Orientações possíveis

Caso	**NTP**	**NMV**	**OP**
Albano	24	7,25	0°, 180°
Dagli	30	6,30	0°, 180°
Dighe1	16	3,87	0°
Dighe2	10	4,70	0°
Fu	12	3,58	0°, 90°, 180°, 270°
Jakobs1	25	5,60	0°, 90°, 180°, 270°
Jakobs2	25	5,36	0°, 90°, 180°, 270°
Mao	20	9,22	0°, 90°, 180°, 270°
Marques	24	7,37	0°, 90°, 180°, 270°
Shapes0	43	8,75	0°
Shapes1	43	8,75	0°, 180°
Shapes2	28	6,29	0°, 180°
Shirts	99	6,63	0°, 180°
Trousers	64	5,06	0°, 180°

Tabela 6.9: Resultados dos problemas da literatura resolvidos utilizando o algoritmo primal. **CM**: Comprimento mínimo obtido

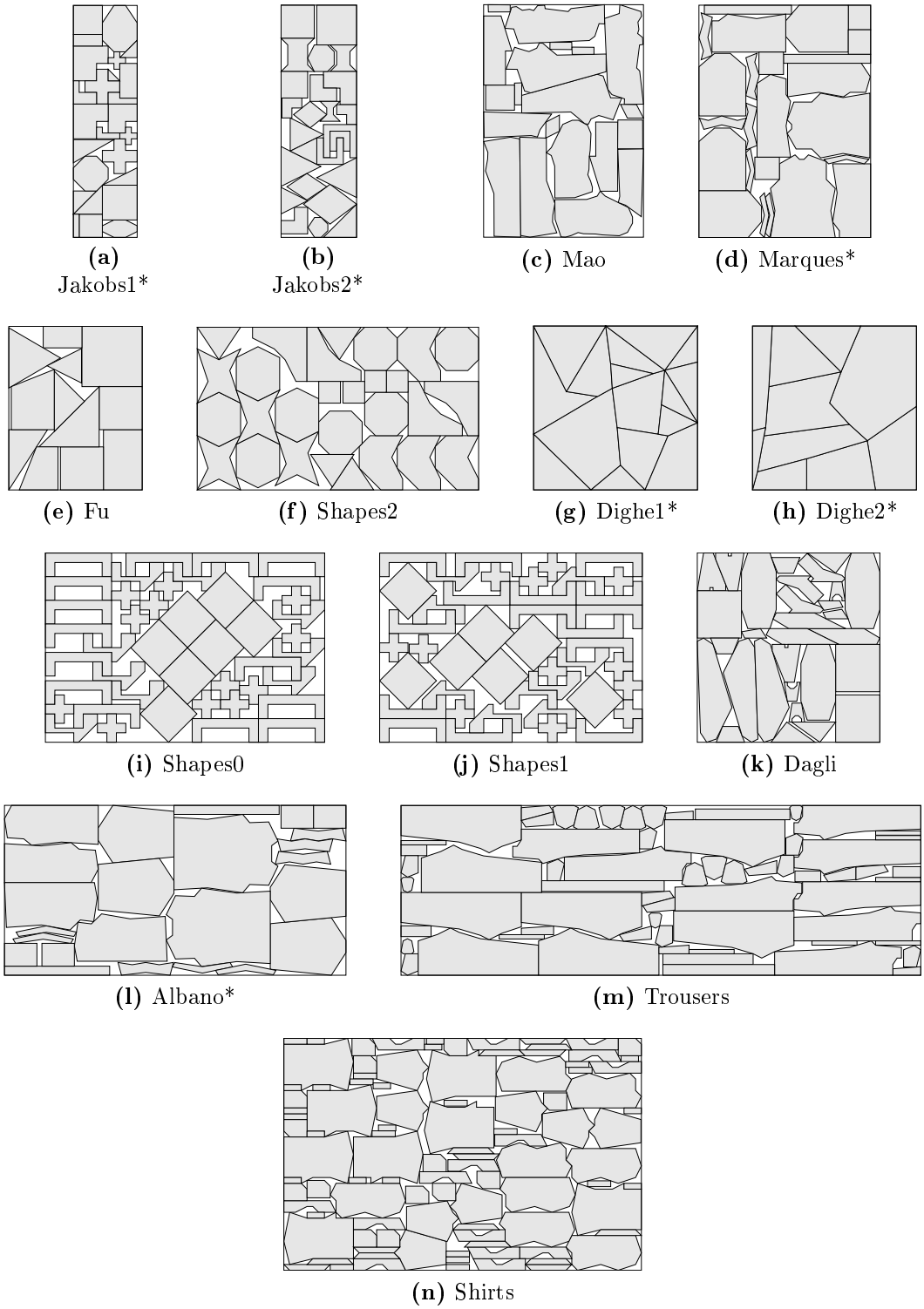
Caso	**CM**	Densidade (%)
Albano	9906,45	87,78
Dagli	58,20	86,97
Dighe1	1000,00	100,00
Dighe2	1000,00	100,00
Fu	30,97	92,02
Jakobs1	11,00	89,09
Jakobs2	24,00	80,42
Mao	1767,00	83,61
Marques	77,30	89,49
Shapes0	60,00	66,50
Shapes1	55,00	72,55
Shapes2	26,51	81,48
Shirts	62,00	87,10
Trousers	242,50	89,82

ner. Os parâmetros de redução do comprimento contêiner p_{dim} e de expansão p_{aum} adotados foram 1% e $0,3\%$ respectivamente, resultando em uma redução maior do que a expansão. Estes parâmetros permaneceram fixos durante toda a execução, e foram adotados os mesmos parâmetros para todos os casos. A Tabela 6.10 apresenta os menores comprimentos obtidos e a densidade dos leiautes com esta estratégia. Também são indicados os melhores resultados publicados na literatura. Os leiautes obtidos pelo algoritmo proposto para os casos Albano, Jakobs2 e Marques são mais compactos do que os melhores resultados publicados na literatura. Já o problema Jakobs1 igualou o resultado de [7]. O algoritmo também foi capaz de resolver os dois problemas que possuem solução ótima com densidade de 100%: Dighe1 e Dighe2.

Comparando as duas estratégias, é possível notar que o algoritmo dual desenvolvido obteve resultados melhores na maior parte dos casos, com exceção do problema Fu e Shapes1. Os problemas Dighe1, Dighe2 e Jakobs1 apresentaram um comprimento mínimo igual utilizando as duas estratégias.

Tabela 6.10: Resultados dos problemas da literatura resolvidos utilizando o algoritmo dual. **CM**: Comprimento mínimo obtido. I: [18]. B: [49]. E: [7] G: [6]. Os resultados marcados com * são os melhores obtidos

Caso	Algoritmo proposto		Melhor solução na literatura	
	CM	Densidade (%)	**CM**	Densidade (%)
Albano	**9848,72**	**88,39***	9874,48 (I)	88,16
Dagli	57,82	87,71	**57,63** (B)	**87,87***
Dighe1	**1000**	**100***	**1000 (BG)**	**100***
Dighe2	**1000**	**100***	**1000 (BG)**	**100***
Fu	30,99	91,96	**30,97 (E)**	**92,03***
Jakobs1	**11,00**	**89,07***	**11,00 (E)**	**89,07***
Jakobs2	**22,75**	**84,83***	23,39 (I)	82,51
Mao	1753,20	84,07	**1731,26 (E)**	**85,15***
Marques	**76,85**	**90,01***	77,04 (E)	89,82
Shapes0	59,03	67,59	**58,30 (I)**	**68,44***
Shapes1	55,51	71,88	**54,04 (EI)**	**73,84***
Shapes2	25,93	83,30	**25,64 (I)**	**84,25***
Shirts	61,65	87,59	**60,18 (B)**	**89,69***
Trousers	241,83	90,07	**241,23 (E)**	**90,46***



(a) Jakobs1*
(b) Jakobs2*
(c) Mao
(d) Marques*
(e) Fu
(f) Shapes2
(g) Dighe1*
(h) Dighe2*
(i) Shapes0
(j) Shapes1
(k) Dagli
(l) Albano*
(m) Trousers
(n) Shirts

Figura 6.7: As melhores soluções encontradas pelo algoritmo dual proposto. Os leiautes marcados com * são os melhores obtidos.

6.3 Resultados da paralelização

Antes de se realizar os testes em paralelo, é necessário determinar qual o método serial mais rápido para se obter uma base de comparação. Como visto na seção 3.4, a sequência escolhida para as operações de união e subtração influencia no desempenho do cálculo da região livre de colisão. Assim, foram adotados dois métodos: o método das subtrações e o método das uniões e subtração. Para cada um dos casos da literatura testados na seção anterior, foi escolhido um item móvel e definido um subconjunto de polígonos de obstrução transladados. Tanto o item como o subconjunto de polígono de obstrução foram definidos aleatoriamente, no entanto, estes representam uma situação possível de posicionamento de um item. Este conjunto é utilizado para se calcular a região livre de colisão.

Cada um dos testes constituiu 1000 repetições do cálculo da região livre de colisão e o tempo total de execução do algoritmo é medido. Os tempos obtidos podem ser vistos na Tabela 6.11, nas colunas do método que utiliza subtrações e do método de uniões e subtração serial. O gráfico de barras (Fig. 6.8), mostra de forma mais clara que o método das subtrações é mais lento apenas nos casos Shirts e Trousers.

Tabela 6.11: Geração da região livre de colisão repetida 1000 vezes. **NPO**: número de polígonos de obstrução; **SUB**: tempo de processamento (s) utilizando apenas subtrações (utiliza apenas um núcleo); **US**: tempo de processamento (s) utilizando uniões e subtração

Caso		Método				
			US			
	NPO	**SUB**	serial	2 núcleos	3 núcleos	4 núcleos
Albano	20	2,33	3,13	2,66	2,47	2,48
Dagli	27	2,20	2,53	2,01	1,83	1,65
Dighe1	9	0,63	1,03	0,96	0,98	0,94
Dighe2	6	0,58	0,93	0,89	0,93	0,93
Fu	9	0,62	0,85	0,88	0,79	0,78
Jakobs1	21	1,54	2,37	1,95	1,87	1,71
Jakobs2	21	1,91	2,34	1,98	1,89	1,75
Mao	16	1,77	2,31	2,08	2,08	1,98
Marques	19	2,23	3,24	2,64	2,55	2,54
Shapes0	37	4,24	4,57	3,43	3,11	3,05
Shapes1	40	4,38	4,66	3,44	2,96	2,68
Shapes2	23	1,39	2,06	1,73	1,63	1,52
Shirts	94	12,53	6,54	4,47	3,95	3,41
Trousers	57	12,00	6,93	5,02	4,88	4,74

A interface de programação de aplicações (API) OpenMP foi utilizada para

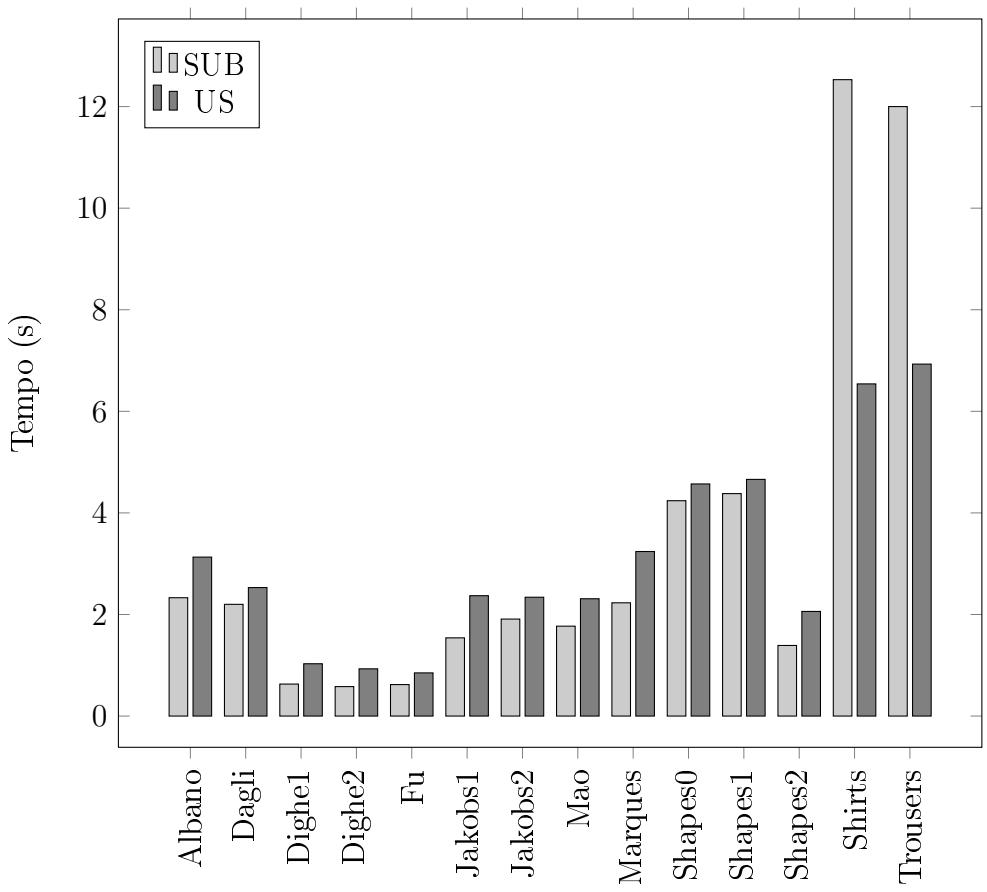


Figura 6.8: Tempo de execução do algoritmo serial para a determinação da região livre de colisão. SUB: método das subtrações. US: método das uniões e subtração.

desenvolver o algoritmo paralelizado. Como o processador testado possui apenas quatro núcleos, este foi o limite adotado para os testes. Os resultados podem ser verificados também na Tabela 6.11. Os problemas Dighe1, Dighe2, Fu, Jakobs1, Mao, Marques e Shapes2 tiveram tempo de execução maior no algoritmo paralelo, independente do número de núcleos utilizados, quando comparado com o menor tempo obtido por um dos algoritmos seriais.

A Fig. 6.9 apresenta a comparação dos tempos de execução dos algoritmos serial e paralelo para obter a região livre de colisão em cada caso da literatura. O algoritmo serial escolhido para cada caso é o que apresenta o menor tempo computacional. Desta forma, apenas para os problemas Trousers e Shirts foi adotado o método das uniões e subtração para algoritmo serial.

A Fig. 6.9a mostra o tempo para os três casos que possuem menos polígonos de obstrução. Nestes casos, o algoritmo paralelo apresenta tempos muito superiores do que o serial. No segundo grupo de problemas (Fig. 6.9c), Dagli sempre apresenta tempos menores para o algoritmo paralelo enquanto que o problema Jakobs2 é mais rápido utilizando paralelização para mais de dois núcleos. No en-

tanto, mesmo utilizando quatro núcleos, há um ganho de desempenho de apenas $1,09$ vezes. Os demais casos representados neste gráfico não se beneficiam da paralelização. Os casos da Fig. 6.9b são os que apresentam os maiores ganhos de desempenho ao realizar a paralelização da determinação da região livre de colisão, podendo chegar a $1,91$ vezes na solução do Shirts utilizando quatro núcleos.

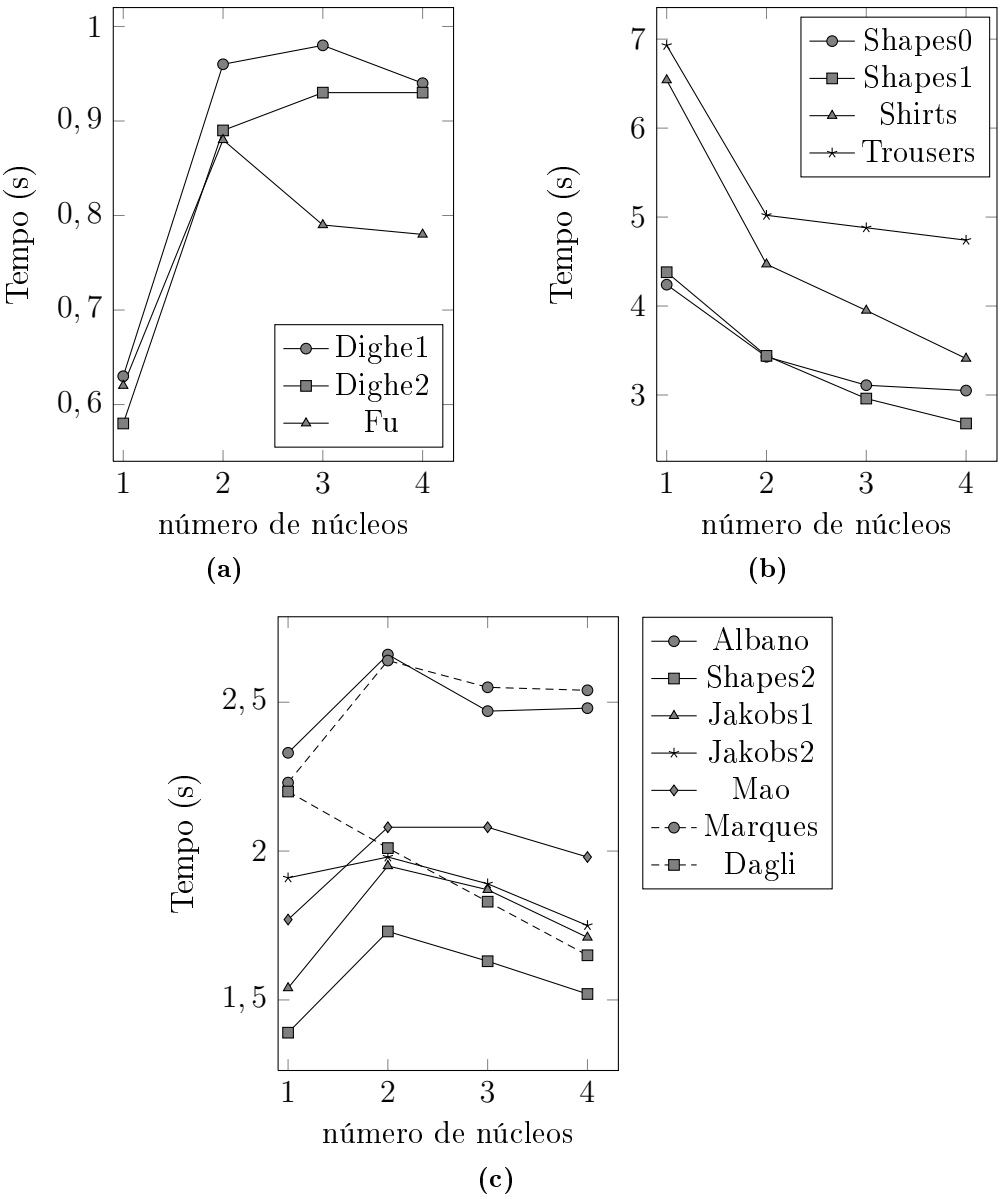


Figura 6.9: Tempos de execução para o algoritmo paralelo. O tempo obtido utilizando apenas um núcleo é o menor tempo obtido por um dos algoritmos seriais propostos.

7 Conclusões

Este trabalho aborda dois problemas de empacotamento de itens irregulares distintos: o problema primal, em que é realizado o posicionamento rotacional em um contêiner de dimensões fixas, e o problema dual que consiste em posicionar itens em um contêiner retangular com uma dimensão variável. O algoritmo adota a estratégia de representar a solução como uma sequência de itens e aplica uma regra de posicionamento para construir a solução. Para cada item, é calculada a sua região livre de colisão, que representa todas as translações possíveis para inserir o item no contêiner que possui outros itens já posicionados. Para o cálculo da região livre de colisão, é necessário empregar operações Booleanas não regularizadas.

Com emprego das operações Booleanas não regularizadas, foi possível obter pontos onde o posicionamento de um item constitui um encaixe perfeito. Estes pontos foram priorizados no posicionamento, constituindo uma heurística que, em conjunto com a limitação do posicionamento apenas nos vértices da região livre de colisão, auxiliou o recozimento simulado na busca pelo ótimo global, obtendo menores tempo de execução e número de iterações para convergir. Além disso, vértices côncavos da região livre de colisão foram ignorados. Novos estudos podem ser realizados para descobertas de novas propriedades do posicionamento na região livre de colisão. Para os problemas duais, foi possível observar que a adoção do nível externo não só contribuiu para automatizar o processo, mas também para obter leiautes mais compactos. A grande diferença entre estas duas abordagens é que, no caso do algoritmo dual proposto, a solução atual é sempre aproveitada na próxima execução do nível interno. Isto teve grande influência na obtenção de leiautes mais densos. O estudo dos métodos de determinação da região livre de colisão mostrou que a ordem das operações Booleanas têm impacto no desempenho do algoritmo e varia de acordo com o número de vértices e intersecções. Uma análise mais profunda é necessária para avaliar a influência desses fatores, além da sua relação com a geometria dos itens e contêineres. Este estudo pode indicar estratégias adequadas a serem empregadas para cada problema, com

o objetivo de maximizar o desempenho, mesmo no caso serial.

Também visando melhorar o desempenho geral do algoritmo, outras medidas também podem ser tomadas. O desenvolvimento de uma heurística de posicionamento que realiza a busca por ótimos locais é uma abordagem popular. A combinação com o recozimento simulado já foi introduzida por [6]. Entretanto, é possível utilizar o conceito de região livre colisão, determinada através de operações Booleanas não regularizadas, para se determinar estes ótimos locais, que são as posições de encaixe. Outra possibilidade é a utilização de um algoritmo de compactação que, após a construção do leiaute, verifica os graus de liberdade de cada item e os movimenta a fim de acomodar outro item, também com o auxílio da região livre de colisão. Desta forma, o conceito de região livre de colisão, apesar de apresentar custo computacional elevado, ainda pode ser melhor explorado para se obter uma convergência mais rápida do algoritmo.

Referências

1 WÄSCHER, G.; HAUSSNER, H.; SCHUMANN, H. An improved typology of cutting and packing problems. *European Journal of Operational Research*, v. 183, p. 1109–1130, 2007.

2 BABU, A. R.; BABU, N. R. A generic approach for nesting of 2d parts in 2d sheets using generic and heuristic algorithms. *Computer Aided Design*, v. 33, p. 879–891, 2001.

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

4 BURKE, E. K.; HELLIER, R. S. R.; KENDALL, G.; WHITWELL, G. Complete and robust no–fit polygon generation for the irregular stock cutting problem. *European Journal of Operational Research*, v. 179, p. 27–49, 2007.

5 BENNELL, J. A.; DOWSLAND, K. A. A tabu thresholding implementation for the irregular stock cutting problem. *Int. J. Prod. Res.*, v. 37, p. 4259–4275, 1999.

6 GOMES, A. M.; OLIVEIRA, J. F. Solving irregular strip packing problems by hybridising simulated annealing and linear programming. *European Journal of Operational Research*, v. 171, p. 811–829, 2006.

7 EGEBLAD, J.; NIELSEN, B. K.; ODGAARD, A. Fast neighborhood search for two- and three–dimensional nesting problems. *European Journal of Operational Research*, v. 183, p. 1249–1266, 2007.

8 HECKMANN, R.; LENGAUER, T. A simulated annealing approach to the nesting problem in the textile manufacturing industry. *Annals of Operations Research*, v. 57, p. 103–133, 1995.

9 MARTINS, T. C.; TSUZUKI, M. S. G. Simulated annealing applied to the irregular rotational placement of shapes over containers with fixed dimensions. *Expert Systems with Applications*, v. 37, p. 1955–1972, 2010.

10 FOWLER, R. J.; PATERSON, M.; TANIMOTO, S. L. Optimal packing and covering in the plane are np-complete. *Inf. Process. Lett.*, v. 12, n. 3, p. 133–137, 1981.

11 JAKOBS, S. On genetic algorithms for the packing of polygons. *European Journal of Operational Research*, v. 88, p. 165–181, 1996.

12 BENNELL, J. A.; DOWSLAND, K. A. Hybridising tabu search with optimisation techniques for irregular stock cutting. *Manage. Sci.*, v. 47, p. 1160–1172, 2001.

13 LI, Z.; MILENKOVIC, V. Compaction and separation algorithms for non-convex polygons and their applications. *European Journal of Operations Research*, v. 84, p. 539–561, 1995.

14 STOYAN, Y. G.; NOVOZHILOVA, M. V.; KARTASHOV, A. V. Mathematical model and method of searching for a local extremum for the non-convex oriented polygons allocation problem. *European Journal of Operational Research*, v. 92, n. 1, p. 193 – 210, 1996.

15 DANIELS, K.; MILENKOVIC, V. J. Column-based strip packing using ordered and compliant containment. *In Proc. 1st ACM Workshop on Appl. Comput. Geom*, p. 33–38, 1996.

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

17 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.

18 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.

19 KIRKPATRICK, S.; GELLAT, C. D.; VECCHI, M. P. Optimization by simulated annealing. *Science*, v. 220, p. 671–680, 1983.

20 CORANA, A.; MARCHESI, M.; MARTINI, C.; RIDELLA, S. Minimizing multimodal functions of continuous variables with the simulated annealing algorithm. *ACM Transactions on Mathematical Software*, v. 13, p. 262–280, 1987.

21 MARTINS, T. C.; TSUZUKI, M. S. G. Placement over containers with fixed dimensions solved with adaptive neighborhood simulated annealing. *Bulletin of the Polish Academy of Sciences Technical Sciences*, v. 57, p. 273–280, 2009.

22 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, n. 2, p. 371 – 381, 2002.

23 ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. *NBR 100004*: Resíduos sólidos. Rio de Janeiro, 2005.

24 MARTINS, T. C. *Estudo do Recozimento Simulado e do Polígono de Obstrução Aplicados ao Problema de Empacotamento Rotacional de Polígonos Irregulares Não–Convexos em Recipientes Fechados*. Tese (Doutorado) — Universidade de São Paulo, 2007.

25 SEGENREICH, S. A.; BRAGA, L. M. P. F. Optimal nesting of general plane figures: A monte carlo heuristical approach. *Computers and Graphics*, v. 10, n. 3, p. 229 – 237, 1986.

26 BENNELL, J. A.; OLIVEIRA, J. F. The geometry of nesting problems: A tutorial. *European Journal of Operational Research*, v. 184, n. 2, p. 397–415, 2008.

27 MARTINS, T. C.; TSUZUKI, M. S. G. Rotational placement of irregular polygons over containers with fixed dimensions using simulated annealing and no-fit polygons. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, v. 30, p. 205 – 212, 09 2008.

28 ART, R. C. An approach to the two-dimensional irregular cutting stock problem. *Technical Report 36.008, IBM Cambridge Centre*, 1966.

29 MAHADEVAN, A. *Optimization in computer-aided pattern packing (marking, envelopes)*. Tese (Doutorado) — North Carolina State University, 1984. AAI8507009.

30 AGARWAL, P. K.; FLATO, E.; HALPERIN, D. Polygon decomposition for efficient construction of minkowski sums. *Computational Geometry*, v. 21, p. 39–61, 2002.

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

32 SATO, A. K.; TAKIMOTO, R. Y.; MARTINS, T. C.; TSUZUKI, M. S. G. Translational placement using simulated annealing and collision free region with parallel processing. *Proceedings of the 9th IEEE/IAS International Conference on Industry Applications*, São Paulo, 2010.

33 HOBBY, J. D. Practical segment intersection with finite precision output. *Computational Geometry*, v. 13, p. 199–214, 1999.

34 SATO, A. K.; TAKIMOTO, R. Y.; MARTINS, T. C.; TSUZUKI, M. S. G. Translational placement using simulated annealing and collision free region with parallel processing. *Anais do XVIII Congresso Brasileiro de Automática*, Bonito, 2010.

35 GONG, J. H.; ZHANG, H.; ZHANG, G. F.; SUN, J. G. Solid reconstruction using recognition of quadric surfaces from orthographic views. *Computer Aided Design*, v. 38, p. 821–835, 2006.

36 YAN, Q. W.; CHEN, C. L. P.; TANG, Z. Efficient algorithm for the reconstruction of 3d objects from orthographic projections. *Computer Aided Design*, v. 26, p. 699–717, 1994.

37 VATTI, B. R. A generic solution to polygon clipping. *Communications of the ACM*, v. 35, p. 56–63, 1992.

38 ZALIK, B.; GOMBOSI, M.; PODGORELEC, D. A quick intersection algorithm for arbitrary polygons. *SCCG98 Conf. On Computer Graphics and its Applications*, p. 195–204, 1998.

39 LIU, Y. K.; WANG, X. Q.; BAO, S. Z.; GOMBOSI, M.; ZALIK, B. An algorithm for polygon clipping, and for determining polygon intersections and unions. *Computer & Geosciences*, v. 33, p. 589–598, 2007.

40 LEONOV, M. V.; NIKITIN, A. G. An efficient algorithm for a closed set of boolean operations on polygonal regions in the plane. *A. P. Ershov Institute of Informatics Systems, Preprint 46*, 1997.

41 BENTLEY, J. L.; OTTMANN, T. A. Algorithms for reporting and counting geometric intersections. *IEEE Transactions on Computers*, C-28, p. 643–647, 1979.

42 SHAMOS, M. I.; HOEY, D. Geometric intersection problems. *Foundations of Computer Science, 1976., 17th Annual Symposium on*, p. 208 –215, 1976.

43 M'HALLAH, R.; BOUZIRI, A.; JILANI, W. Layout of two dimensional irregular shapes using genetic algorithms. *Lecture Notes In Computer Science*, v. 2070, p. 403–411, 2001.

44 MILENKOVIC, V. Rotational polygon overlap minimization and compaction. *Computational Geometry*, v. 10, p. 305–318, 1998.

45 ADAMOWICZ, M.; ALBANO, A. Nesting two-dimensional shapes in rectangular modules. *Computer-Aided Design*, v. 8, n. 1, p. 27 – 33, 1976.

46 METROPOLIS, N.; ROSENBLUTH, A.; ROSENBLUTH, M.; TELLER, A.; TELLER, E. Equation of state calculations by fast computing machines. *J. Chem. Phys.*, v. 21, p. 1087, 1953.

47 SATO, A. K.; MARTINS, T. C.; TSUZUKI, M. S. G. Rotational placement using simulated annealing and collision free region. *10th IFAC Workshop on Intelligent Manufacturing Systems (Preprints)*, Lisboa, p. 253–258, 2010.

48 MARTINS, T. C.; TSUZUKI, M. S. G. Comparison of deterministic heuristics and simulated annealing for the rotational placement problem over containers with fixed dimensions. *Proceedings of the IFAC Symposium Information Control Problems in Manufacturing*, Moscow, 2009.

49 BENNELL, J. A.; SONG, X. A comprehensive and robust procedure for obtaining the no-fit polygon using minkowski sums. *European Journal of Operational Research*, v. 35, p. 267–281, 2008.