ANDRÉ KUBAGAWA SATO

Solução Rasterizada para o Problema de Empacotamento de Fita
Irregular Utilizando a Montanha Voronoi

São Paulo
2015

ANDRÉ KUBAGAWA SATO

Solução Rasterizada para o Problema de Empacotamento de Fita
Irregular Utilizando a Montanha Voronoi

Tese apresentada à Escola Politécnica da
Universidade de São Paulo para obtenção
do título de Doutor em Ciências

São Paulo
2015

ANDRÉ KUBAGAWA SATO

Solução Rasterizada para o Problema de Empacotamento de Fita
Irregular Utilizando a Montanha Voronoi

Tese apresentada à Escola Politécnica da
Universidade de São Paulo para obtenção
do título de Doutor em Ciências

Área de Concentração:
Engenharia Mecânica

Orientador: Prof. Dr. Marcos de Sales
Guerra Tsuzuki

Coorientador: Prof. Dr. Thiago De Castro
Martins

São Paulo
2015

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, _____ de _____ de _____

Assinatura do autor: _____

Assinatura do orientador: _____

Catálogo-na-publicação

Sato, André Kubagawa

Solução Rasterizada para o Problema de Empacotamento de Fita Irregular Utilizando a Montanha Voronoi / A. K. Sato -- versão corr. -- São Paulo, 2015. 112 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecânica.

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

A meus avós

Agradecimentos

Agradeço ao 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 ao Prof. Dr. Thiago de Castro Martins pelo grande auxílio no desenvolvimento da pesquisa.

Agradeço à minha família pelo apoio e incentivo e a minha parceira de vida, pelo suporte incondicional.

Agradeço a FAPESP (processo 2010/19646-0), pelo apoio financeiro e institucional.

*“Genius might be the ability
to say a profound thing
in a simple way.”
Charles Bukowski*

Resumo

O empacotamento irregular de fita é um grupo de problemas na área de corte e empacotamento, cuja aplicação é observada nas indústrias têxtil, moveleira e construção naval. O problema consiste em definir uma configuração de itens irregulares de modo que o comprimento do contêiner retangular que contém o leiaute seja minimizado. A solução deve ser válida, isto é, não deve haver sobreposição entre os itens, que não devem extrapolar as paredes do contêiner. Devido a aspectos práticos, são admitidas até quatro orientações para o item. O volume de material desperdiçado está diretamente relacionado à qualidade do leiaute obtido e, por este motivo, uma solução eficiente pressupõe uma vantagem econômica e resulta em um menor impacto ambiental. O objetivo deste trabalho consiste na geração automática de leiautes de modo a obter níveis de compactação e tempo de processamento compatíveis com outras soluções na literatura. A fim de atingir este objetivo, são realizadas duas propostas de solução. A primeira consiste no posicionamento sequencial dos itens de modo a maximizar a ocorrência de posições de encaixe, que estão relacionadas à restrição de movimento de um item no leiaute. Em linhas gerais, várias sequências de posicionamentos são exploradas com o objetivo de encontrar a solução mais compacta. Na segunda abordagem, que consiste na principal proposta deste trabalho, métodos rasterizados são aplicados para movimentar itens de acordo com uma grade de posicionamento, admitindo sobreposição. O método é baseado na estratégia de minimização de sobreposição, cujo objetivo é a eliminação da sobreposição em um contêiner fechado. Ambos os algoritmos foram testados utilizando o mesmo conjunto de problemas de referência da literatura. Foi verificado que a primeira estratégia não foi capaz de obter soluções satisfatórias, apesar de fornecer informações importantes sobre as propriedades das posições de encaixe. Por outro lado, a segunda abordagem obteve resultados competitivos. O desempenho do algoritmo também foi compatível com outras soluções, inclusive em casos nos quais o volume de dados era alto. Ademais, como trabalho futuro, o algoritmo pode ser estendido de modo a possibilitar a entrada de itens de geometria genérica, o que pode se tornar o grande diferencial da proposta.

Palavras-chave: Empacotamento de fita. Região livre de colisão. Método rasterizado.

Abstract

Irregular nesting belongs to the area of cutting and packing problems and are employed in the textile, wood and shipbuilding industries. The problem consists in determining a configuration for a set of irregular items which minimizes the length of the rectangular container in which the layout is located. The solution must be feasible, i.e., items must not overlap nor protrude the container walls. Due to practical reasons, up to four orientations are allowed for an item. The volume of wasted material is directly affected by the quality (density) of the layout. Thus, an efficient solution produces a positive economic and environmental impact. In this work, the objective is to automatically obtain layouts such that their density and the performance of the algorithm are competitive with other solutions in literature. So as to achieve this goal, two approaches are proposed. The first method uses a special sequential placement heuristic such that the algorithm maximizes exact placements, which consist of constrained positions for items. In general terms, a search is performed in the placement sequence in order to obtain a compact layout. In the second approach, which is the main subject of this work, raster methods are employed to guide the translation of items, which are free to move within the layout, and may overlap other items. The method is based on overlap minimization techniques, in which the objective is to eliminate the overlap in a fixed dimensions container. Both algorithms were tested using benchmark problems from the literature. The first strategy yielded unsatisfactory results, though it provided important information about the properties of exactly fitting placements. On the other hand, the main approach was able to produce competitive solutions. The performance was also compatible with other solutions, even in cases which the data volume was high. Moreover, as a future work, an extension for the algorithm can be developed such that items with generic geometry can be considered, which would be an important advance in research terms.

Keywords: Nesting problem. Collision free region. Raster method.

Lista de ilustrações

Figura 1 – Exemplo de leiaute para indústria têxtil	13
Figura 2 – Tipologia para problemas de empacotamento	18
Figura 3 – Problemas básicos encontrados na literatura	19
Figura 4 – Método rasterizado para detectar sobreposição	20
Figura 5 – Polígono de obstrução obtido através do método de escorregamento	22
Figura 6 – Propriedades do polígono de obstrução	22
Figura 7 – Operadores para modificar a posição de um item	26
Figura 8 – Heurística inferior esquerda	30
Figura 9 – Problema de empacotamento de fita	34
Figura 10 – Problema adaptado para contêiner fixo	35
Figura 11 – Mapeamento de translações fornecendo o polígono de obstrução	38
Figura 12 – Polígono de posicionamento interno	39
Figura 13 – Obtenção do polígono de posicionamento interno	40
Figura 14 – Leiaute válido obtido utilizando a região livre de colisão	41
Figura 15 – Determinação da região livre de colisão	42
Figura 16 – Posição de encaixe deslizante	43
Figura 17 – Posição de encaixe perfeito	43
Figura 18 – Obtenção de aresta degenerada	46
Figura 19 – Obtenção de vértice degenerado	46
Figura 20 – Criação da lista ordenada de arestas	47
Figura 21 – Obtenção das listas direta e indireta	48
Figura 22 – Obtenção do setor a partir das listas direta e indireta	49
Figura 23 – Determinação completa das listas direta e indireta	49
Figura 24 – Comparação de um setor do polígono de obstrução com os setores da região livre de colisão da Figura 23	50
Figura 25 – Determinação de uma aresta da região degenerada através do método de escorregamento	51
Figura 26 – Itens e contêiner para o caso exemplo	53
Figura 27 – Polígono de obstrução para o caso exemplo	53
Figura 28 – Região livre de colisão para o caso exemplo	53
Figura 29 – Comparação de arestas para o caso exemplo	53
Figura 30 – Comparação de setores para o caso exemplo	54
Figura 31 – Região degenerada para o caso exemplo	54
Figura 32 – Restrição da região degenerada para o caso exemplo	54
Figura 33 – Exemplos de posicionamento para o caso exemplo	55
Figura 34 – Heurísticas de posicionamento de par com priorização de encaixe	57

Figura 35 – Compactação do leiaute sem admissão de sobreposição	57
Figura 36 – Leiautes mais compactos obtidos com o algoritmo baseado em posições de encaixe	65
Figura 37 – Problema de minimização de sobreposição	68
Figura 38 – Área de sobreposição entre itens	71
Figura 39 – Exemplo de profundidade de penetração	71
Figura 40 – Diagrama de Voronoi de um polígono convexo.	73
Figura 41 – Montanha Voronoi de um polígono convexo	74
Figura 42 – Polígono de obstrução rasterizado	76
Figura 43 – Montanha de penetração discreta	76
Figura 44 – Passos para a obtenção da montanha de penetração discreta	77
Figura 45 – Troca de dois itens no leiaute	78
Figura 46 – Exemplo de mapa de sobreposição total	80
Figura 47 – Sequência de execuções da busca local	81
Figura 48 – Fuga do ótimo local utilizando a busca local guiada	83
Figura 49 – Compactação do leiaute com a admissão de sobreposição	84
Figura 50 – Obtenção da posição de menor sobreposição de um item utilizando a busca com duas resoluções	87
Figura 51 – Modelos de vetor do algoritmo 4SED	89
Figura 52 – Exemplo de uma varredura do algoritmo 4SED	90
Figura 53 – Modelos de vetor do algoritmo paralelo	90
Figura 54 – Ganhos de processamento da transformada distância para o caso Albano	93
Figura 55 – Ganhos de processamento para a geração do mapa de sobreposição total (Jakobs2)	95
Figura 56 – Comparação das soluções das variantes do algoritmo proposto (maior escala)	98
Figura 57 – Comparação das soluções das variantes do algoritmo proposto (menor escala)	99
Figura 58 – Comparação dos resultados com as soluções da literatura (densidades médias)	102
Figura 59 – Comparação dos resultados com as soluções da literatura (densidades máximas)	103
Figura 60 – Leiautes mais compactos obtidos através do algoritmo baseado em métodos rasterizados (com sobreposição)	104
Figura 61 – Leiautes mais compactos obtidos através do algoritmo baseado em métodos rasterizados (sem sobreposição)	105

Lista de tabelas

Tabela 1 – Dados dos problemas de referência da literatura	60
Tabela 2 – Resultados do algoritmo baseado em posições de encaixe (um nível) . .	61
Tabela 3 – Resultados do algoritmo baseado em posições de encaixe (dois níveis) .	62
Tabela 4 – Comparação dos resultados do algoritmo baseado em posições de encaixe	62
Tabela 5 – Comparação entre as densidades médias obtidas pelo algoritmo baseado em posições de encaixe e por outras soluções	63
Tabela 6 – Melhores soluções obtidas em comparação com as melhores na literatura	64
Tabela 7 – Resultados da paralelização da transformada distância	93
Tabela 8 – Resultados da paralelização da geração do mapa de sobreposição total	94
Tabela 9 – Resultados do algoritmo baseado em métodos rasterizados	96
Tabela 10 – Densidades médias (em porcentagem) obtidas por soluções na literatura comparadas às obtidas pelo algoritmo baseado em métodos rasterizados	100
Tabela 11 – Densidades máximas (em porcentagem) obtidas por soluções na li- teratura comparadas às obtidas pelo algoritmo baseado em métodos rasterizados	101
Tabela 12 – Comparação das duas propostas deste trabalho	101

Sumário

1	INTRODUÇÃO	12
1.1	Objetivo e contribuições	14
1.2	Estrutura do texto	15
2	EMPACOTAMENTO IRREGULAR DE FITA	16
2.1	Tipologia de problemas de empacotamento	16
2.2	Geometria de problemas com itens irregulares	19
2.2.1	Método rasterizado	20
2.2.2	Função D	21
2.2.3	Polígono de obstrução	21
2.2.4	Função Phi	23
2.3	Aspectos gerais das soluções na literatura	24
2.3.1	Representação da geometria	24
2.3.2	Conectividade do espaço de busca	25
2.3.3	Busca pela optimalidade	26
2.3.4	Definição da função objetivo	27
2.4	Revisão bibliográfica	28
2.4.1	Soluções matemáticas	28
2.4.2	Busca na sequência	29
2.4.3	Busca no leiaute	31
3	SOLUCIONANDO O EMPACOTAMENTO IRREGULAR DE FITA UTILIZANDO O ENCAIXE PERFEITO	33
3.1	Definição do problema	33
3.2	Breve histórico	35
3.3	Conceitos geométricos	37
3.3.1	Polígono de obstrução	37
3.3.2	Polígono de posicionamento interno	38
3.3.3	Região livre de colisão	40
3.3.4	Posições de encaixe	43
3.4	Obtenção das posições de encaixe	44
3.4.1	Regras para obtenção de aresta e vértices degenerados	44
3.4.2	Região degenerada	45
3.4.3	Determinação de arestas de mesma orientação	47
3.4.4	Algoritmo para comparação de setores	47
3.4.5	Construção da região degenerada	51

3.4.6	Restrição do posicionamento	52
3.4.7	Estudo de caso	52
3.5	Aplicação no problema de empacotamento irregular de fita	54
3.5.1	Heurísticas de posicionamento	55
3.5.2	Compactação do leiaute	56
3.5.3	Heurística de busca	58
3.6	Resultados	59
4	ABORDAGEM RASTERIZADA PARA O PROBLEMA DE MINI- MIZAÇÃO DE SOBREPOSIÇÃO	66
4.1	Discussão sobre a questão da rasterização	66
4.2	Problema de minimização de sobreposição	68
4.3	Deteccção e avaliação de sobreposição	69
4.3.1	Função sobreposição	69
4.3.2	Transformada distância	75
4.4	Minimização de sobreposição	77
4.4.1	Mapa de sobreposição total	78
4.4.2	Descrição da busca local	79
4.4.3	Busca local guiada	81
4.5	Compactação do leiaute	82
4.5.1	Solução inicial	84
4.6	Extensão do método	85
4.6.1	Abordagem com múltiplas resoluções	86
4.6.2	Paralelização massiva	87
4.7	Resultados	92
4.7.1	Paralelização da geração da montanha de penetração discreta	92
4.7.2	Paralelização da obtenção do mapa de sobreposição total	94
4.7.3	Algoritmo de empacotamento de fita	95
5	DISCUSSÃO DOS RESULTADOS	106
6	CONCLUSÃO	108
	REFERÊNCIAS	109

1 Introdução

Problemas de corte e empacotamento constituem uma importante área em pesquisa operacional. Este grupo reúne essencialmente casos em que objetos devem ser alocados em contêineres ou padrões devem ser recortados de algum material. Apesar de distintas, essas duas classes de problemas podem ser modeladas matematicamente de forma similar, tornando as soluções intercambiáveis. Neste trabalho, é investigado um problema específico, o empacotamento irregular de fita, que consiste em posicionar itens em um contêiner retangular de comprimento variável de modo a minimizar o comprimento do leiaute. Os itens, que são representados por polígonos, podem ser irregulares e admitem um número finito de orientações, que em geral se limitam aos ângulos 0° , 90° , 180° ou 270° . As principais restrições são: não deve existir sobreposição entre os itens e todos devem estar completamente inseridos no contêiner. Este é um problema recorrente na indústria têxtil, embora também seja encontrado nas indústrias moveleira, naval, de vidro e na construção naval.

A indústria têxtil é o maior foco em pesquisas de empacotamento irregular de fita e define as restrições geralmente aplicadas a trabalhos no tema. O problema pode ser enunciado como: dado um rolo grande de tecido, de largura conhecida, definir o padrão de recortes de modo a reduzir o desperdício de material quando uma sequência de operações de corte é realizada ao longo do comprimento do rolo. A [Figura 1](#) mostra um exemplo de uma solução para um caso da indústria têxtil. Originalmente, os itens podem ser compostos de linhas, arcos de circunferência ou até outras curvas, mas são geralmente aproximados por polígonos para processamento computacional. O número de orientações admissíveis para os itens é determinado pela presença de estampas, diferenciação dos lados do tecido e/ou direção das fibras.

Uma solução eficiente para um problema de corte ou de empacotamento pode gerar um grande impacto econômico e, muitas vezes, ambiental. Considere, por exemplo, um desafio clássico da indústria moveleira: a definição de cortes retos em placas de madeira a fim de obter as peças que serão montadas em um produto final. Neste problema de corte, na maior parte das vezes haverá volume de madeira não aproveitada, uma vez que as placas possuem dimensões predefinidas. O estudo deste problema de corte visa reduzir o desperdício de material, que pode diminuir o custo total de produção, caso seja possível reduzir o número de placas utilizadas. Neste caso, há também um menor impacto ambiental, uma vez que uma menor quantidade de matéria prima é consumida. Um exemplo simples de problema na área de empacotamento consiste no transporte de cargas em caminhões. Considerando que a frota de veículos possui uma capacidade de carga muito grande, uma solução ótima minimiza o número de caminhões que serão utilizados

Figura 1 – Exemplo de leiaute para indústria têxtil

Fonte: o autor

para realizar o transporte, reduzindo o gasto de combustível e o custo total da operação.

Problemas de corte e empacotamento podem ser resolvidos satisfatoriamente por especialistas humanos. No entanto, um grande impedimento é a grande variação do tempo que uma pessoa pode levar para obter uma configuração eficiente, desde poucas horas até muitos dias. Outro problema é o viés muitas vezes observado no comportamento humano para um determinado modo de resolver o problema, que nem sempre é a maneira mais eficiente. Por exemplo, no caso de problemas com itens irregulares, é comum observar que humanos tendem a posicionar o item na posição inferior esquerda. Entretanto, é possível obter maiores taxas de compactação através de técnicas mais avançadas. Desta forma, as melhores soluções podem não ser intuitivas para um especialista.

Isto posto, existe uma extensa literatura de soluções computadorizadas para o problema de empacotamento irregular de fita. O problema pode ser visualizado por dois aspectos: o geométrico e o de otimização. Os itens são irregulares, o que significa que não estão limitados a formas básicas como retângulos ou círculos, gerando uma complexidade maior para garantir a condição de não sobreposição entre os itens. Já do ponto de vista de problema de otimização, uma solução deve buscar pela melhor configuração dos itens de modo a reduzir o material desperdiçado. É possível notar que a busca está condicionada à definição da geometria, uma vez que o leiaute final deve ser necessariamente válido (sem sobreposição). Assim, os avanços na literatura são verificados em duas frentes: no desenvolvimento de ferramentas geométricas mais robustas e eficientes e na aplicação de métodos de otimização que gerem leiautes mais compactos e/ou em menor tempo.

Na literatura, é possível observar dois modos de realizar a otimização do leiaute. O primeiro consiste em buscar uma sequência de modo que, ao posicionar os itens um a um seguindo uma determinada regra, a solução final seja a mais compacta possível. O segundo consiste em movimentar os itens livremente até encontrar um leiaute válido que seja o mais eficiente. Ambas as soluções são difíceis de elaborar. No primeiro caso, existe o problema combinatório na busca da sequência de itens, além da complexidade de encontrar um conjunto de regras que gere um posicionamento mais compacto. No outro caso, que admite posicionamento livre, as possibilidades de movimentação dos itens são infinitas e, ademais, não é trivial garantir a validade do leiaute dada tamanha liberdade de posicionamento. Devido a essas dificuldades, a maior parte das propostas utiliza heurísticas na busca de uma solução eficiente, não necessariamente ótima, para o problema. Encontrar uma boa combinação entre uma ferramenta geométrica e uma heurística de busca é geralmente o objetivo na elaboração de soluções para o empacotamento irregular de fita.

1.1 Objetivo e contribuições

O objetivo do trabalho é obter uma solução para o problema de empacotamento irregular de fita através de uma abordagem heurística. Assim, não será visado obter o ótimo global, porém as soluções devem ser suficientemente compactas. A fim de verificar a qualidade dos leiautes obtidos, serão realizados testes com problemas de referência, cujos resultados serão comparados com as melhores soluções da literatura.

Soluções heurísticas podem ser definidas por duas características fundamentais: o modo como lidam com a geometria irregular e o método de busca adotado para encontrar a melhor solução. A resolução destes dois problemas deve ser feita de modo que elas se favoreçam. Neste trabalho, são propostas duas soluções para o empacotamento irregular de fita. Cada uma das alternativas foi elaborada de modo a realizar acoplamentos diferentes e inovadores para as questões da geometria e da otimização.

A primeira proposta utiliza o algoritmo de [Sato, Martins e Tsuzuki \(2012\)](#) para determinar posições de encaixe para itens representados através de polígonos. Apesar da aparente importância deste tipo de posicionamento, o seu potencial ainda não foi investigado e comprovado. Neste trabalho, é proposto um algoritmo que insere itens no contêiner exclusivamente em posições de encaixe a fim de estudar as vantagens e desvantagens destas posições. Um algoritmo de recozimento simulado é adotado para realizar a exploração da sequência de posicionamento e obter um algoritmo capaz de resolver os problemas de referência eficientemente.

Para a segunda proposta, o posicionamento de um item no leiaute é realizado em coordenadas discretas, definidas a partir da aplicação de uma grade. Esta técnica visa reduzir o espaço de exploração na vizinhança do posicionamento de um item e, neste

trabalho, é aplicada pela primeira vez a uma abordagem de minimização de sobreposição. A abordagem discreta implica na realização de operações matriciais, que podem ser aceleradas a partir da paralelização massiva de operações e, conseqüentemente, melhorar o desempenho do algoritmo. Estas contribuições fornecem ferramentas e conceitos inéditos que podem ser incorporados a novas soluções, além de apresentar mais duas alternativas para a solução de problemas de empacotamento irregular de fita.

1.2 Estrutura do texto

Este texto é estruturado da seguinte forma. O [Capítulo 2](#) descreve aspectos importantes do problema e das soluções de empacotamento irregular de fita. A [seção 2.1](#) descreve brevemente a tipologia mais aceita para problemas de empacotamento e define a categoria na qual o problema estudado se enquadra. Em seguida, na [seção 2.2](#), são apresentadas as principais ferramentas geométricas utilizadas para resolver as restrições de sobreposição. A [seção 2.3](#) faz um resumo das principais características das soluções propostas e uma revisão bibliográfica é apresentada na [seção 2.4](#). Os dois capítulos seguintes descrevem as soluções propostas neste trabalho para a solução do problema de empacotamento irregular de fita. O [Capítulo 3](#) descreve a primeira proposta realizada, que consiste em priorizar posições de encaixe de itens no leiaute. A segunda e principal proposta descrita neste texto está contida no [Capítulo 4](#), que utiliza ferramentas rasterizadas para solucionar o problema de minimização de sobreposição. Os resultados de cada proposta são apresentados nas respectivas seções e, no [Capítulo 5](#), é realizada uma discussão conjunta dos resultados obtidos pelos dois algoritmos desenvolvidos. O [Capítulo 6](#) apresenta as conclusões e algumas sugestões de trabalhos futuros para estender as pesquisas realizadas neste trabalho.

2 Empacotamento irregular de fita

O problema de empacotamento irregular de fita é uma variante do grupo de problemas de corte e empacotamento, que constituem uma importante área em pesquisa operacional. A principal característica, do ponto de vista das soluções desenvolvidas na literatura, é a geometria dos itens, que pode ser irregular. Desta forma, soluções requerem um processamento mais complexo para satisfazer as restrições geométricas, o que impacta a busca do leiaute mais compacto, que, por sua vez, constitui o objetivo principal do problema.

Neste capítulo, as características gerais e possíveis variações do problema são analisadas a partir do estudo da tipologia de problemas de corte e empacotamento proposta por [Wäscher, Haußner e Schumann \(2007\)](#). Em seguida, as ferramentas geométricas utilizadas para lidar com a detecção de sobreposição em problemas com itens irregulares são apresentadas. Estes tópicos servem como base para a descrição e análise das propostas da literatura para resolução de problemas de empacotamento irregular de fita.

2.1 Tipologia de problemas de empacotamento

Na prática, problemas de corte são geralmente bastantes distintos de problemas de empacotamento. Entretanto, do ponto de vista teórico, os dois grupos podem assumir uma mesma formulação. [Dyckhoff \(1990\)](#) salientou esta estrutura compartilhada entre os problemas ao elaborar uma tipologia conjunta para os problemas de corte e empacotamento. Esta abordagem facilitou o intercâmbio e fusão de soluções entre os dois grupos. Entretanto, devido a problemas linguísticos, nos quais se inclui a utilização de siglas a partir de palavras em alemão, esta tipologia não recebeu aceitação universal na literatura. Além destes problemas, a adoção da tipologia de [Dyckhoff \(1990\)](#) sofreu resistência devido à classificação inconsistente de problemas atuais. Por estes motivos, [Wäscher, Haußner e Schumann \(2007\)](#) propuseram uma nova tipologia para problemas de corte e empacotamento, que é descrita no restante desta seção.

Os problemas de corte e empacotamento podem ser representados de forma unificada. Dados dois conjuntos de elementos: um conjunto de objetos grandes e um conjunto de itens pequenos; selecione um grupo de itens pequenos, separe em um ou mais subgrupos e, por fim, associe cada subgrupo a um objeto grande de modo que as condições geométricas sejam respeitadas. As condições geométricas, comuns a todos os problemas de corte e empacotamento são:

- todos os itens pequenos devem estar completamente contidos em um objeto grande,

e

- não deve existir sobreposição entre os itens pequenos.

Foram definidos cinco critérios para agrupar os problemas: dimensionalidade, tipo de atribuição, variedade dos itens pequenos, variedade dos objetos grandes e a forma dos itens pequenos. Os critérios são brevemente descritos a seguir.

- Dimensionalidade – uma, duas ou três dimensões. Representam a dimensão mínima capaz de representar a geometria dos itens e do contêiner.
- Tipo de atribuição – maximização do valor da saída ou minimização do valor da entrada. No primeiro caso, conjuntos de itens pequenos devem ser associados a conjuntos de objetos grandes, de modo que todos os objetos grandes sejam utilizados. Já no caso de minimização do valor de entrada, cada item pequeno deve ser associado a pelo menos um objeto grande.
- Variedade dos itens pequenos – os itens podem ser idênticos, de variedade fracamente heterogênea ou de variedade fortemente heterogênea.
- Variedade dos objetos grandes – pode existir um ou mais objetos grandes. No caso de objeto único, as dimensões podem ser fixas ou variáveis. Nos demais casos, o conjunto deve ser classificado como: objetos grandes idênticos, de variedade fracamente heterogênea ou de variedade fortemente heterogênea.
- Forma dos itens pequenos – nos casos bidimensional ou tridimensional, os itens podem ser classificados como irregulares ou regulares.

A [Figura 2](#) agrupa as possíveis classificações para cada grupo segundo a tipologia de [Wäscher, Haußner e Schumann \(2007\)](#). Uma importante modificação introduzida nesta tipologia é a identificação clara de problemas em que o objeto grande possui uma ou mais dimensões variáveis. Problemas que não se enquadram em nenhuma classe são considerados variantes do problema. Além disso, são considerados problemas estendidos os casos que tratam de aspectos adicionais que fogem à área de corte e empacotamento, como, por exemplo, sequência de processamento ou tamanho de lote.

Problemas básicos são definidos a partir dos critérios de tipo de atribuição e variedade dos itens pequenos a fim de propiciar uma nomenclatura que seja mais facilmente aceita. Os tipos básicos de problemas possíveis estão ilustrados na [Figura 3](#). A definição adicional da variedade dos objetos grandes resulta em um problema intermediário. Se forem acrescentadas as características de dimensionalidade e forma dos itens pequenos, um problema refinado é definido. Uma pesquisa científica geralmente se concentra em um problema refinado, acrescido de restrições adicionais.

Figura 2 – Tipologia para problemas de empacotamento

Fonte: o autor

O problema estudado neste trabalho se encaixa no grupo de problemas de dimensão variável (*open dimension problems*). Ademais, apenas um objeto grande com uma dimensão variável é adotado, o que define o problema intermediário. Por fim, é estudado o caso bidimensional com itens pequenos irregulares. A fim de facilitar a compreensão do texto, nas demais seções, o termo item é sempre utilizado em referência a item pequeno, enquanto o objeto grande é denominado contêiner.

Figura 3 – Problemas básicos encontrados na literatura

Fonte: Wäscher, Haußner e Schumann (2007)

2.2 Geometria de problemas com itens irregulares

O primeiro desafio encontrado na solução de problemas de empacotamento irregular de fita é a dificuldade em tratar a geometria complexa dos itens. Este obstáculo é um dos motivos do reduzido número de trabalhos na literatura na área de itens irregulares (BEN-NELL; OLIVEIRA, 2008). Desta forma, a fim de possibilitar a solução do problema, foram estudadas na literatura algumas ferramentas geométricas específicas. Garantir que os itens estejam totalmente contidos no contêiner é uma tarefa relativamente simples, uma vez que, no caso de empacotamento de fita, os contêineres são sempre retangulares. Desta forma, para um problema de empacotamento, a ferramenta geométrica mais básica deve ser capaz de informar, para um par de itens, se os mesmos estão separados ou em sobreposição. Os aspectos mais importantes são a precisão e o tempo computacional necessário para verificar a sobreposição. Quatro métodos são discutidos: o método rasterizado, a função D, o polígono de obstrução e a função phi.

2.2.1 Método rasterizado

Uma das formas mais simples de lidar com geometrias complexas é adotar uma representação rasterizada (*raster method*), também conhecida como representação por pixels. Os itens são representados por matrizes cujos valores são atribuídos de forma a distinguir a região interna da externa. A [Figura 4](#) apresenta um exemplo de um item representado através de pixels. Quando as arestas não são verticais ou horizontais, é possível observar no exemplo que a representação não corresponde totalmente à geometria original. Este erro gerado pode ser reduzido através do refinamento da grade utilizada para a rasterização. Entretanto, quanto menor o tamanho da grade, maior o volume de informação gerada e, conseqüentemente, mais elevados são os requisitos de memória para o armazenamento da geometria e maior o tempo para processar o volume de dados.

Figura 4 – Método rasterizado para detectar sobreposição

Fonte: o autor

A detecção de sobreposição utilizando a representação rasterizada pode ser feita de diversas formas, as quais podem ser basicamente reduzidas a uma operação entre matrizes. Um dos métodos mais simples para checar se os itens se sobrepõem consiste em atribuir o valor 1 para posições internas e 0 para a região externa. Quando um item é posicionado, a matriz é somada com as matrizes dos demais itens no leiaute e posições com valores maiores que 1 correspondem a regiões de sobreposição (vide [Figura 4](#)). [Segenreich e Braga](#)

(1986) expandiram a proposta utilizando 3 valores, sendo que um deles é atribuído para o contorno do item, possibilitando a checagem de posições em que itens se encostam. Uma implementação mais avançada foi proposta por Babu e Babu (2001), que utiliza os valores da matriz para facilitar a definição da menor translação horizontal necessária para separar os itens.

A representação rasterizada é bastante versátil, capaz de representar e detectar a sobreposição entre itens com geometria genérica. A maior desvantagem é a falta de precisão do método em comparação a outros. Dependendo da implementação, o leiaute final pode apresentar sobreposição ou haverá espaço entre os itens, o que pode comprometer a qualidade da solução. A precisão pode ser controlada a partir do ajuste dos parâmetros da rasterização. No entanto, além do acréscimo no uso de memória, a detecção de sobreposição é mais custosa à medida que o tamanho das matrizes aumenta.

2.2.2 Função D

A trigonometria direta pode ser utilizada para solucionar o problema de determinação da sobreposição. Neste caso, é adotada a representação poligonal dos itens, o que implica uma maior precisão do método. Deste modo, é possível desenvolver um método exato, entretanto, aspectos práticos de implementação limitam a precisão do algoritmo. O modo direto de detecção de sobreposição consiste em comparar os retângulos envolventes dos itens e, de acordo com o resultado, realizar a checagem direta de intersecção entre todos os pares de arestas.

Bennell e Oliveira (2008) propuseram uma série de testes para tornar o método de trigonometria direta mais eficiente. Para verificar a intersecção entre arestas, a função D foi utilizada. A função D descreve a posição relativa de um ponto em relação a uma reta. Mahadevan (1984) apresentou um método que utiliza a função D para determinar eficientemente a posição relativa de duas arestas.

A função D pode ser utilizada para determinar com precisão a sobreposição entre itens. O problema principal é a necessidade da realização dos cálculos de intersecção toda vez que a posição do item é modificada. Assim, sua aplicação é bastante custosa para algoritmos de empacotamento iterativos, que realizam movimentações de itens a cada iteração.

2.2.3 Polígono de obstrução

O polígono de obstrução (*no-fit polygon*) foi introduzido por Art (1966) e é a ferramenta geométrica mais empregada em problemas de empacotamento com itens irregulares devido à sua precisão e desempenho. A precisão é obtida a partir da utilização da representação geométrica e o desempenho é assegurado a partir da não necessidade de

calcular o polígono de obstrução mais de uma vez para cada par de itens.

O método de escorregamento (*sliding algorithm*) de obtenção do polígono de obstrução proposto por Mahadevan (1984) é um dos mais intuitivos e de fácil compreensão. Por este motivo, ele será utilizado para demonstrar as propriedades do polígono de obstrução. Considere dois itens com orientações fixas. Um item é denominado item fixo, que, durante o processo de determinação do polígono de obstrução, permanece imóvel. O outro item do par é chamado de item orbital, ou item móvel. É necessário definir um ponto de referência para o item móvel a priori. Inicialmente, os itens devem estar posicionados de modo que eles se encostem. O contorno do polígono de obstrução é então definido pelo caminho que o ponto de referência do item orbital traça à medida que este desliza ao redor do item fixo. A Figura 5 apresenta um exemplo do método de escorregamento para a determinação do polígono de obstrução.

Figura 5 – Polígono de obstrução obtido através do método de escorregamento

Fonte: o autor

Figura 6 – Propriedades do polígono de obstrução

Fonte: o autor

A partir da Figura 6, é possível observar as principais propriedades no polígono de obstrução. Considerando que se deseja verificar se os dois itens se sobrepõem, três situações são verificadas:

- o ponto de referência está localizado no interior do polígono de obstrução – os itens estão em sobreposição;
- o ponto de referência é externo ao polígono de obstrução – os itens estão separados;
- o ponto de referência está localizado em algum ponto do contorno do polígono de obstrução – os itens se encostam.

Desta forma, o polígono de obstrução mapeia as posições relativas proibidas para um par de itens a partir da definição de um ponto de referência. A identificação de sobreposição é reduzida essencialmente a uma operação de determinação da posição relativa de um ponto em um polígono. Isto é válido uma vez que todos os polígonos de obstrução do problema, um para cada par, já estão definidos, o que pode ser realizado em uma etapa de pré-processamento.

O polígono de obstrução é relativamente fácil de obter quando os itens são convexos. No caso de itens côncavos, existe um desafio em definir um algoritmo robusto para o seu cálculo. Diversas soluções foram propostas na literatura, como: extensão do algoritmo de deslizamento ([WHITWELL, 2005](#); [BURKE et al., 2007](#)), soma de Minkowski ([BENNEL; DOWSLAND; DOWSLAND, 2001](#); [BENNEL; SONG, 2008](#)) e decomposição em polígonos convexos ([AGARWAL; FLATO; HALPERIN, 2002](#)).

2.2.4 Função Phi

A função Phi é um conceito que visa descrever todas as posições entre dois objetos a partir de expressões matemáticas, que são essencialmente obtidas a partir de funções de distância. O polígono de obstrução pode ser considerado um caso especial de função Phi que só fornece informação limitada. Na proposta de [Stoyan et al. \(2001\)](#), a função Phi foi aplicada para problemas de corte e empacotamento.

O valor da função Phi é maior que zero nas configurações em que os itens estão separados, zero quando os itens se encostam e menor que zero no caso de sobreposição. Quando a função é normalizada, seu valor corresponde à distância euclidiana. Uma das maiores dificuldades para sua aplicação decorre do fato de que não existe um método eficiente para a determinação da função Phi para geometrias arbitrárias.

Uma vez que formulações para formas primárias, como círculos ou retângulos, e seus complementos já foram elaboradas ([STOYAN et al., 2001](#)), é possível obter a função Phi para outras geometrias a partir da decomposição em formas primárias. Utilizando este método, [Chernov et al. \(2012\)](#) elaboraram uma metodologia para obter a função Phi para objetos que podem ser representados através de linhas e arcos de circunferência.

Apesar do seu potencial, devido a complexidade da sua definição, a adoção da função Phi ainda é muito restrita. A vantagem da função Phi é sua capacidade de fornecer

mais informação a respeito da configuração dos itens. Entretanto, ainda não foi comprovado o ganho que se pode obter ao adotar esta ferramenta.

2.3 Aspectos gerais das soluções na literatura

O desenvolvimento de uma solução para o empacotamento irregular de fita passa por uma série de definições, cada uma capaz de impactar a eficiência e o desempenho do algoritmo final. Na literatura, não existe um consenso sobre o melhor método para resolver o problema de empacotamento irregular de fita e é possível encontrar métodos muitos distintos capazes de obter resultados similares. As principais definições que devem ser realizadas para se elaborar uma solução, baseadas nas propostas existentes da literatura, são discutidas nesta seção. Aspectos mais específicos de determinada classe de soluções são abordados na [seção 2.4](#), onde é feita uma revisão bibliográfica do tema.

2.3.1 Representação da geometria

A primeira escolha a ser feita é a definição do método de representação da geometria dos itens e do contêiner. Isso afetará a disponibilidade de ferramentas geométricas que podem ser utilizadas para realizar a detecção de sobreposição no leiaute.

O contêiner em problemas básicos de empacotamento de fita é retangular e, desta forma, geralmente não é necessário adotar uma representação mais complexa. Usualmente, o contêiner é representado apenas pelo valor de sua largura, uma vez que o comprimento é a dimensão variável.

No problema de empacotamento irregular de fita, os itens podem ser irregulares. Neste trabalho, um item é considerado irregular quando não é possível descrever a sua geometria com apenas um parâmetro. Desta forma, círculos e polígonos regulares não são considerados irregulares. O conceito pode se estender a geometrias tridimensionais, porém isto está fora do escopo deste trabalho. A fim de aplicar um algoritmo de empacotamento irregular, é desejável que exista ao menos um polígono irregular. Caso contrário, a solução pode ser obtida mais eficientemente utilizando uma solução específica para a geometria regular desejada.

Existem dois tipos básicos de representação para os itens: rasterizada ou poligonal. A ferramenta geométrica depende da representação adotada. O impacto da escolha recai principalmente na precisão na detecção da sobreposição, nos requisitos de memória necessária para executar o algoritmo e no desempenho computacional.

A representação rasterizada utiliza uma matriz para descrever a geometria de um item. Esta matriz é obtida a partir da definição de uma grade de tamanho predefinido. Posições da grade que contém alguma região interna ao item são marcadas com um valor

diferente. A representação rasterizada possui a vantagem de ser capaz de representar itens de geometria genérica e existem ferramentas geométricas simples para lidar com sobreposições (ver [subseção 2.2.1](#)). Entretanto a baixa precisão obtida é a principal desvantagem desta abordagem. Caso desejado, uma maior precisão pode ser obtida ao custo de um tempo computacional mais elevado para verificar sobreposições.

A utilização de polígonos para representar a geometria é denominada representação poligonal ou vetorial. Nesta representação, os itens são representados por uma lista de vetores de duas dimensões, que por sua vez correspondem às coordenadas dos vértices do polígono. Em geral a lista é ordenada de modo que o interior do polígono se localize sempre à esquerda da aresta. Quando um item não puder ser descrito com exatidão utilizando polígonos, a representação poligonal deve ser aproximada, o que gera uma imprecisão no método. Apesar disso, soluções que adotam a representação poligonal geralmente possuem maior precisão. Como desvantagem, é possível salientar que os cálculos são mais complexos. No entanto, o desempenho não depende do tamanho da grade como no caso rasterizado. Devido a sua precisão, a grande maioria das soluções na literatura utiliza a representação vetorizada. A representação poligonal pode ser estendida incorporando outras geometrias básicas como arcos de circunferência, como em ([BURKE et al., 2010](#)). Neste caso, ferramentas geométricas específicas devem ser adotadas ou desenvolvidas.

2.3.2 Conectividade do espaço de busca

A solução de um problema de empacotamento consiste em um conjunto de vetores, que representam o posicionamento de cada item, e um conjunto de orientações. Para realizar a busca, existem essencialmente duas representações para a solução: a representação direta em coordenada de cada item ou a sequência de posicionamento dos itens. No segundo caso, é necessária uma heurística construtiva que gera um leiaute válido a partir da sequência de entrada.

O espaço de soluções é considerado conexo quando uma pequena perturbação na entrada causa modificações pequenas na solução final. A partir desta afirmação, é possível definir o conceito de vizinhança de uma solução, que consiste no conjunto de soluções próximas no espaço de soluções, que são geradas a partir da alteração mínima de apenas um parâmetro. Em geral, algoritmos de busca são mais eficientes quando o espaço de busca é mais conexo.

Quando a sequência de itens é adotada como representação do leiaute, o espaço de soluções é usualmente bastante desconexo. Isto pode ser observado na [Figura 7](#), que apresenta casos em que uma pequena mudança na ordem de posicionamento ocasiona um grande impacto no leiaute. Quando as coordenadas dos itens são controladas diretamente, é possível verificar uma maior conectividade do espaço de busca. Entretanto, é grande a probabilidade de gerar leiautes inválidos se nenhuma técnica especial é empregada para

determinar os parâmetros de posição.

Figura 7 – Operadores para modificar a posição de um item

Fonte: [Bennell e Oliveira \(2009\)](#)

Na representação direta nas posições dos itens, quando um item é movimentado, a vizinhança da busca é infinita, uma vez que o espaço é contínuo. Por este motivo, a busca pode ser bastante ineficiente. Algumas soluções na literatura contornam este problema adotando uma grade para o posicionamento de itens ([TOLEDO et al., 2013](#); [BENNELL; DOWSLAND, 1999](#)), o que torna o espaço de busca finito. Apesar de reduzir um pouco a conectividade do espaço de soluções, os algoritmos de buscas ainda são muito efetivos quando empregados nestes casos. A maior desvantagem é a impossibilidade de se obter soluções mais compactas devido à limitação quanto aos posicionamentos dos itens.

2.3.3 Busca pela optimalidade

O problema de empacotamento irregular de fita, assim como outros problemas de empacotamento mais simples, é NP-completo ([FOWLER; PATERSON; TANIMOTO, 1981](#)). Desta forma, não é possível obter uma solução ótima para todos os problemas de empacotamento irregular em tempo computacional aceitável. Por este motivo, é muitas

vezes conveniente optar por abordagens heurísticas para obter uma boa solução para o problema, embora não comprovadamente ótima.

Soluções matemáticas para problemas de empacotamento são capazes de obter ótimos globais para problemas específicos, com um número reduzido de itens e sem considerar rotação (ALVAREZ-VALDES; MARTINEZ; TAMARIT, 2013; TOLEDO et al., 2013). O tempo computacional para solucionar casos mais complexos cresce bastante, tornando a solução impraticável. É muito importante a obtenção da solução ótima para casos padrões na literatura, uma vez que todos os demais métodos podem utilizar esta informação como referência. Entretanto, no caso de problemas de empacotamento irregulares de fita, não foi comprovada ainda a optimalidade de nenhuma solução para algum problema de referência. Outra função importante de soluções matemáticas é a capacidade de obtenção de ótimos locais. Métodos híbridos combinam essa capacidade de obtenção de soluções ótimas localmente com buscas heurísticas, buscando movimentar vários itens ao mesmo tempo (GOMES; OLIVEIRA, 2006; IMAMICHI; YAGIURA; NAGAMOCCHI, 2009).

Soluções heurísticas são adotadas na obtenção de soluções compactas para problemas de empacotamento irregular de fita. Dentre as abordagens heurísticas, é possível separar duas estratégias: as que trabalham com a construção de uma única solução e as que modificam uma solução pronta iterativamente para se obter compactação incremental. O primeiro grupo geralmente utiliza uma heurística construtiva para garantir a geração de um leiaute válido. A ordem de posicionamento dos itens pode ser fixa ou dinâmica. No caso de sequência dinâmica, a definição da ordem é realizada a partir da avaliação parcial da solução. Já o segundo grupo geralmente utiliza algum método de busca local. Os mais populares são: o recozimento simulado (SATO; MARTINS; TSUZUKI, 2012; GOMES; OLIVEIRA, 2006), a busca tabu (BENNELL; DOWSLAND, 1999; BLAZEWICZ; HAWRYLUK; WALKOWIAK, 1993) e os algoritmos genéticos (JAKOBS, 1996; BABU; BABU, 2001).

2.3.4 Definição da função objetivo

O problema de empacotamento irregular de fita pode ser modelado como a minimização do comprimento do leiaute sujeito às restrições geométricas. Neste caso existe apenas uma função a ser minimizada, que pode ser definida como o comprimento do leiaute ou a área desperdiçada no contêiner. Entretanto, é possível relaxar a restrição de sobreposição entre os itens de modo a tornar o espaço de busca mais conexo. Neste caso, temos dois objetivos a serem alcançados: eliminar a sobreposição e reduzir o comprimento do leiaute. A maior dificuldade desta abordagem é obter leiautes válidos ao fim da execução, uma vez que ótimos locais geralmente apresentam sobreposição entre itens.

Na literatura, existem dois modos principais para se realizar a minimização quando os dois objetivos são contemplados. O primeiro método constitui na atribuição de pesos

aos componentes das funções, buscando obter uma solução válida. Os pesos podem ser dinâmicos, mudando ao longo das iterações (HECKMANN; LENGAUER, 1995) ou estáticos. A segunda estratégia trabalha com um objetivo de cada vez, adotando um contêiner de dimensões fixas. Assim, a eliminação da sobreposição do leiaute constitui o objetivo principal. Uma vez obtido o leiaute válido, o contêiner pode ser reduzido de modo incremental. Neste método, o problema de minimização de sobreposição deve ser resolvido muitas vezes, o que constitui sua principal desvantagem. Além disso, o comprimento inicial do contêiner influencia bastante no desempenho da solução.

2.4 Revisão bibliográfica

Vários fatores são considerados no desenvolvimento de uma solução para o problema de empacotamento irregular de fita. A maior parte dos aspectos discutidos pode ser combinada, gerando resultados bastante distintos, mesmo com uma pequena mudança em apenas uma das características que compõe a solução. Isto posto, é difícil elaborar uma classificação para as soluções na literatura, uma vez que cada proposta pode se encaixar em diversas categorias. Neste trabalho, as abordagens na literatura são categorizadas em três grupos: soluções matemáticas, soluções com busca na sequência e soluções com busca no leiaute. É importante verificar que as últimas duas classificações se referem a soluções que adotam heurísticas para resolver o problema.

2.4.1 Soluções matemáticas

Este grupo representa as soluções na literatura que buscam pelo ótimo global do problema. Devido à complexidade do problema em questão, existem poucas abordagens nesta categoria. Entretanto, é importante destacar que soluções matemáticas que resolvem subproblemas são mais frequentes e muitas vezes servem como base para soluções completas do problema de empacotamento. Por este motivo, uma breve discussão destes métodos é inserida nesta subseção.

Li e Milenkovic (1995) utilizaram um modelo de programação linear capaz de obter um ótimo local se a posição relativa entre itens e a orientação são fixas. A compactação é obtida através da aplicação de pequenos deslocamentos aos itens de modo que o comprimento do leiaute seja reduzido. O modelo utiliza a seguinte propriedade: o ponto de referência de cada item deve estar localizado à direita das arestas dos polígonos de obstrução induzidos pelo item. Todos os itens são movimentados ao mesmo tempo e são necessárias sucessivas aplicações do algoritmo para atingir o mínimo local. O modelo foi modificado a fim de se obter um algoritmo capaz de realizar a separação dos itens através do relaxamento de algumas restrições de sobreposição por Gomes e Oliveira (2006). Imamichi, Yagiura e Nagamochi (2009) utilizaram um modelo de programação não linear

capaz de obter um leiaute que localmente minimiza o valor da sobreposição total a partir de uma formulação sem restrição.

Carravilla, Ribeiro e Oliveira (2003) utilizaram programação lógica por restrições para encontrar a solução ótima de problemas com até sete itens. No campo de programação matemática, Toledo et al. (2013) utilizaram um modelo de programação inteira mista e posicionaram itens em uma grade no contêiner. Esta solução foi capaz de obter o ótimo global para problemas com até 56 itens. É importante notar, no entanto, que a solução ótima para o problema com posicionamento discreto não necessariamente corresponde ao ótimo global do problema original, que pode apresentar uma maior compactação. Baseado no modelo de programação inteira mista proposta em Fischetti e Luzzi (2009), Alvarez-Valdes, Martinez e Tamarit (2013) desenvolveram uma solução que utilizou algoritmo *branch and bound* na busca pela solução ótima do problema. Nesta abordagem, foi proposto um particionamento do espaço externo ao polígono de obstrução, que corresponde a todas as posições possíveis para o posicionamento, em fatias (*slices*), de modo a facilitar a abordagem *branch and bound*. O algoritmo foi capaz de obter a solução ótima para problemas de empacotamento irregular de fita com até 16 itens.

As soluções matemáticas encontradas na literatura são capazes de obter a solução ótima apenas para problemas com um número reduzido de itens. Uma exceção é a solução proposta por Toledo et al. (2013), porém o posicionamento é limitado a pontos discretos. Além disso, para resolver problemas grandes, a variedade dos itens deve ser pequena. As soluções nesta categoria desconsidera a possibilidade de qualquer rotação para os itens, o que limita bastante a aplicabilidade destes algoritmos na prática.

2.4.2 Busca na sequência

Propostas na literatura que utilizam a busca na sequência adotam uma heurística de posicionamento e representam a solução como uma sequência de itens. A heurística de posicionamento, ou heurística construtiva, garante a condição de não sobreposição entre itens durante todo o curso da solução. Desta forma, como não é necessário trabalhar com dois objetivos, apenas a minimização do comprimento do leiaute é considerada. O posicionamento é realizado de forma sequencial, um item por vez, imitando o comportamento humano mais comum nas soluções de problema de empacotamento. Os trabalhos na literatura são caracterizados por dois aspectos principais: a heurística de posicionamento adotada e a estratégia na escolha da sequência de posicionamento.

A estratégia mais utilizada para realizar o posicionamento de um item no leiaute é a inferior esquerda (*bottom-left*). Na sua versão mais simples, o item escolhido é inicialmente posicionado no canto superior direito do contêiner. Em seguida, é deslizado para esquerda até encontrar um obstáculo (outro item ou uma parede do contêiner) e, então, movido para baixo até que seja possível o deslocamento para a esquerda novamente. Estas etapas são

repetidas até que não seja possível mover mais o item. A [Figura 8](#) apresenta um exemplo deste posicionamento.

Figura 8 – Heurística inferior esquerda

Fonte: Adaptado de [Bennell e Oliveira \(2009\)](#)

Quanto à escolha da sequência de posicionamento, existem dois grupos de soluções. O primeiro utiliza uma sequência predefinida, possivelmente obtida aleatoriamente. Já o segundo utiliza algum mecanismo que realiza a escolha dinâmica dos itens. Desta forma, para cada posicionamento, é feita uma análise dos itens que restam para serem posicionados e algum critério é utilizado para escolher o melhor candidato. A dificuldade desta proposta é definir o critério de escolha do próximo item, dado que avaliações parciais muitas vezes não possuem relação com a qualidade final do leiaute.

[Albano e Sapuppo \(1980\)](#) transformam a busca na sequência no problema de encontrar o caminho ótimo em um grafo. A heurística de posicionamento adotada é a inferior esquerda. [Blazewicz, Hawryluk e Walkowiak \(1993\)](#) utilizaram uma heurística de posicionamento que busca por espaços vazios contíguos para o posicionamento. A sequência inicial é obtida pela ordenação decrescente das áreas dos itens, o que intensifica a criação de espaços vazios. A busca tabu é utilizada para aprimorar a solução, realizando troca de posições entre os itens. [Jakobs \(1996\)](#) observou que a utilização do algoritmo genético era natural quando a solução é codificada como uma sequência de itens e, assim, aplicou esta meta-heurística para o problema de empacotamento irregular, utilizando o posicionamento inferior esquerda como heurística construtiva. A mesma combinação de algoritmo genético e heurística inferior esquerda é empregada por [Babu e Babu \(2001\)](#), que utilizam a representação rasterizada para descrever os itens e o contêiner.

[Burke et al. \(2006\)](#) propuseram uma nova heurística construtiva inferior esquerda com capacidade de preencher lacunas no leiaute, que é uma das maiores limitações do método original. O movimento dos itens se inicia no canto inferior esquerdo e o deslizamento horizontal é feito de forma discreta, a partir de uma grade de pontos, enquanto o movimento vertical é contínuo. Para determinar os posicionamentos válidos, o polígono de obstrução é utilizado. Esta técnica é combinada com as buscas locais *hill-climbing* e busca tabu.

Gomes e Oliveira (2002) utilizaram o conceito de polígono de obstrução e geometria computacional para obter diretamente o posicionamento inferior esquerdo, incluindo posições que preenchem lacunas. O posicionamento de um dado item é obtido a partir do cálculo da intersecção entre todos os polígonos de obstrução e o polígono de posicionamento interno. Estas posições são unidas com o conjunto de vértices dos polígonos de obstrução e uma operação de ordenamento é realizada para determinar a posição com o menor valor de coordenada horizontal. Uma heurística *2-exchange* é utilizada para varrer a sequência de modo limitado, a fim de obter maior eficiência.

Oliveira, Gomes e Ferreira (2000) desenvolveram um algoritmo de posicionamento com uma origem flutuante. Desta forma, o posicionamento de um novo item não é limitado horizontalmente, apenas verticalmente. A posição do novo item a ser inserido no leiaute é determinada a partir da escolha de um entre três critérios que visam minimizar o crescimento do comprimento do leiaute. Esta estratégia foi empregada por Bennell e Song (2010), que utilizaram a meta-heurística beam search para realizar a escolha da melhor sequência. Cada posicionamento é representado como um nó em uma estrutura de árvore. A busca é realizada limitando o número de folhas a cada iteração e uma função é definida para avaliar a qualidade de cada nó. Este valor guiará a expansão da árvore, uma vez que é muito custoso realizar a expansão total. As soluções completas são representadas pelos nós de menor altura.

2.4.3 Busca no leiaute

Os algoritmos que realizam a busca no leiaute trabalham diretamente nas coordenadas dos itens e possibilitam sobreposição no leiaute. A maior dificuldade desta abordagem é reduzir simultaneamente a sobreposição e o comprimento do contêiner. No entanto, o espaço de busca é mais conexo que nas propostas de busca na sequência, gerando algoritmos de busca mais eficientes.

Heckmann e Lengauer (1995) abordaram o problema considerando uma função objetivo composta, atribuindo pesos para cada parcela. A busca foi guiada por um algoritmo de recozimento simulado. Estes pesos foram definidos dinamicamente, priorizando a compactação do leiaute no início e, mais próximo do fim, aumentando o peso atribuído à quantidade de sobreposição visando obter um leiaute válido. Todavia, o algoritmo geralmente convergia para um ótimo local com sobreposição. Desta forma, foi proposta uma estratégia para a separação dos itens, também baseada no recozimento simulado.

As demais soluções na literatura utilizam basicamente dois métodos para lidar com o problema de realizar simultaneamente a separação e a compactação do leiaute: 1) métodos de compactação e separação e 2) minimização da sobreposição. Ambos os grupos são descritos a seguir.

1. **Compactação e separação** – Neste grupo de soluções, modelos matemáticos são utilizados para encontrar ótimos locais que minimizam o comprimento do leiaute (compactação) ou eliminam a sobreposição (separação). Estes modelos foram discutidos na [subseção 2.4.1](#), uma vez que muitos servem como base para soluções puramente matemáticas. Algoritmos eficientes para o problema de empacotamento irregular são frequentemente obtidos empregando alguma meta-heurística para realizar a busca por um leiaute mais compacto. Diferentemente de outras propostas que utilizam buscas locais, todos os itens são movidos ao mesmo tempo. [Bennell e Dowsland \(2001\)](#) utilizaram a meta-heurística *tabu thresholding* em combinação com algoritmos de compactação e separação, que foram aplicados quando o valor da sobreposição de uma solução ficava abaixo de um limiar. [Gomes e Oliveira \(2006\)](#) combinaram um algoritmo de recozimento simulado com um modelo de programação linear. A cada iteração, uma alteração é realizada na solução, seguida da execução do algoritmo de separação e, por fim, aplicação do modelo de compactação. O recozimento simulado foi utilizado para evitar que a solução fique presa em um ótimo local.
2. **Minimização de sobreposição** – As soluções deste grupo evitam o problema de analisar um objetivo composto ao fixar o comprimento do contêiner. Desta forma, o único objetivo é eliminar a sobreposição do leiaute. A compactação é obtida a partir da redução incremental do contêiner. Uma das primeiras abordagens foi realizada por [Bennell e Dowsland \(1999\)](#), que propuseram um algoritmo genérico para o problema de empacotamento irregular de fita utilizando a heurística *tabu thresholding*. [Egeblad, Nielsen e Odgaard \(2007\)](#) se basearam no algoritmo descrito em ([FAROE; PISINGER; ZACHARIASEN, 2003](#)), voltado para o problema de *Bin Packing*. A solução proposta para o problema de dimensão aberta analisou, para um dado item, a área de sobreposição ao longo de uma translação horizontal ou vertical. Apesar da continuidade do espaço, este estudo foi viável devido à limitação de apenas um sentido para o movimento. [Umetani et al. \(2009\)](#) ampliaram esta busca, possibilitando translações sequenciais alternadas para os itens e propuseram uma busca local guiada para escapar de ótimos locais. O valor atribuído para a sobreposição foi a profundidade de penetração, que é a medida da menor translação necessária para separar dois itens. [Imamichi, Yagiura e Nagamochi \(2009\)](#) utilizaram um modelo de programação não linear para movimentar todos os itens, encontrando uma configuração que minimiza localmente o valor da sobreposição. [Leung, Lin e Zhang \(2012\)](#) utilizaram este método como ponto de partida e aplicaram a busca *tabu* e um algoritmo de compactação para aprimorar o resultado final. [Elkeran \(2013\)](#) adotou um algoritmo de busca de cuco (*cuckoo search*) para realizar a busca da posição de menor sobreposição no espaço completo dentro do contêiner.

3 Solucionando o empacotamento irregular de fita utilizando o encaixe perfeito

Neste capítulo é descrita a primeira proposta para a solução do problema de empacotamento irregular de fita. Nesta abordagem, a representação poligonal é utilizada, o que fornece boa precisão na detecção de sobreposição entre itens. O foco deste trabalho é a realização de posicionamentos de encaixe, que possuem potencial para gerar leiautes mais compactos. Um item está em uma posição de encaixe quando não é possível movê-lo livremente no leiaute. Para encontrar estas posições, é necessário que os cálculos geométricos possuam alta precisão. Na proposta deste capítulo, o algoritmo de operações Booleanas não regularizadas proposto em (SATO; MARTINS; TSUZUKI, 2013) é adotado para determinar estas posições. Heurísticas de posicionamento específicas são elaboradas e resultados para problemas da literatura são apresentados.

3.1 Definição do problema

O problema de empacotamento irregular de fita considerado para esta primeira abordagem consiste no posicionamento de itens irregulares, admitindo um conjunto de rotações finitas, em um único contêiner retangular que possui uma dimensão variável. O problema pode ser enunciado como: dado um contêiner retangular com uma dimensão definida, um conjunto de n itens (polígonos quaisquer, podendo ser não convexos) e um conjunto de orientações (ângulos para rotação dos itens), determinar a translação e a orientação aplicadas a cada item de modo que os posicione completamente dentro do contêiner, sem sobreposição com os demais, a fim de minimizar a dimensão variável, cujo valor é o mínimo capaz de abrigar todo o leiaute (vide Figura 9).

São fornecidos um contêiner retangular $\mathcal{C} = \mathcal{C}(W, L)$ de largura constante $W > 0$ e comprimento variável $L > 0$, uma lista de polígonos $\mathcal{P} = \{P_1, \dots, P_n\}$, que representam os itens, e uma lista de ângulos $\mathcal{O}$ que corresponde às orientações admissíveis para os itens. Um polígono $P_i \in \mathcal{P}$ rotacionado de o é representado por $P_i(o)$. Por conveniência, o polígono $P_i(o)$ e o contêiner $\mathcal{C}$ representam um conjunto de pontos que inclui as suas regiões internas e contornos.

As translações dos polígonos são representadas por meio de somas de Minkowski. O resultado da soma de Minkowski de um polígono P_i por um vetor de translação $\mathbf{x}_i$, denotada por $P_i \oplus \mathbf{x}_i$, é $\{\mathbf{p} + \mathbf{x}_i \mid \mathbf{p} \in P_i\}$. Para um polígono qualquer P , seja $\imath(P)$ o interior de P .

Figura 9 – Problema de empacotamento de fita

Fonte: o autor

O problema considerado pode ser descrito formalmente como segue:

$$\begin{aligned}
 &\text{minimize} && L \\
 &\text{sujeito a} && \iota(P_i(o_i) \oplus \mathbf{x}_i) \cap (P_j(o_j) \oplus \mathbf{x}_j) = \emptyset \quad 1 \leq i < j \leq n \\
 & && (P_i(o_i) \oplus \mathbf{x}_i) \subseteq \mathcal{C}(W, L) \quad 1 \leq i \leq n \\
 & && L \in \mathbb{R}_+ \\
 & && o_i \in \mathcal{O} \quad 1 \leq i \leq n \\
 & && \mathbf{x}_i \in \mathbb{R}^2 \quad 1 \leq i \leq n.
 \end{aligned} \tag{3.1}$$

A solução do problema ([Equação 3.1](#)) consiste no vetor de translação $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_n)$ e no vetor de orientações $\mathbf{o} = (o_1, \dots, o_n)$. Note que a solução $(\mathbf{x}, \mathbf{o})$ determina unicamente o leiaute dos polígonos. O mínimo comprimento L do contêiner C é formalmente definido pela função

$$\begin{aligned}
 \mu(\mathbf{x}, \mathbf{o}) &= \max\{x \mid (x, y) \in P_i(o_i) \oplus \mathbf{x}_i, P_i \in \mathcal{P}\} - \\
 &\quad \min\{x \mid (x, y) \in P_i(o_i) \oplus \mathbf{x}_i, P_i \in \mathcal{P}\}.
 \end{aligned} \tag{3.2}$$

Nesta primeira abordagem, é considerada a possibilidade de fixar a dimensão do comprimento do contêiner durante a busca pela solução mais compacta, criando um novo subproblema. Este, por sua vez, pode ser enunciado como: dado um contêiner retangular com dimensões fixas, um conjunto de n itens (polígonos quaisquer, podendo ser não convexos) e um conjunto de orientações (ângulos para rotação dos itens), determinar o subconjunto de itens e as transformações (translação e orientação) aplicadas a cada item

de modo que os posicione completamente dentro do contêiner, sem sobreposição com os demais, a fim de maximizar a área ocupada no contêiner (vide Figura 10).

Figura 10 – Problema adaptado para contêiner fixo

Fonte: o autor

Para um polígono qualquer P , seja $A(P)$ sua área. São fornecidos um contêiner retangular $\mathcal{C}$ de dimensões fixas, uma lista de itens $\mathcal{P}$ e uma lista de ângulos $\mathcal{O}$. A lista $\mathcal{E}$ possui os índices dos polígonos que foram posicionados. O objetivo é encontrar o vetor de translações $\mathbf{x}$ e o vetor de orientações $\mathbf{o}$, que são a solução do problema que segue:

$$\begin{aligned} \text{minimize} \quad & A(\mathcal{C}) - \sum_{i \in \mathcal{E}} A(P_i) \\ \text{sujeito a} \quad & \iota(P_i(o_i) \oplus \mathbf{x}_i) \cap (P_j(o_j) \oplus \mathbf{x}_j) = \emptyset \quad i, j \in \mathcal{E}, i \neq j \\ & (P_i(o_i) \oplus \mathbf{x}_i) \subseteq \mathcal{C} \quad i \in \mathcal{E} \end{aligned} \tag{3.3}$$

$$\begin{aligned} & o_i \in \mathcal{O} \quad 1 \leq i \leq n \\ & \mathbf{x}_i \in \mathbb{R}^2 \\ & i \in \mathcal{E}. \end{aligned} \tag{3.4}$$

3.2 Breve histórico

O algoritmo proposto se baseia no conceito de região livre de colisão, que foi introduzido por (MARTINS; TSUZUKI, 2006) e é capaz de descrever todas as posições válidas para o posicionamento de um item. A adoção deste conceito possibilitou representar todo o espaço de configurações válidas a partir da definição de três parâmetros para cada

item. Desta forma, a eficiência do processo de busca pode ser elevado, uma vez que a investigação de soluções inválidas é evitada. Em relação a outros métodos construtivos, a proposta de (MARTINS; TSUZUKI, 2006) possui a vantagem de ampliar o espaço de busca, uma vez que as outras propostas (por exemplo, heurística inferior esquerda) não permitem o controle da posição. A fim de realizar a otimização de leiaute foi empregado um algoritmo de recozimento simulado, cuja capacidade de lidar com um grande número de variáveis, incluindo um misto de variáveis contínuas (posição) e discretas (orientação e sequência), se mostrou apropriada. Em (MARTINS; TSUZUKI, 2009b), o algoritmo de recozimento simulado aplicado ao empacotamento foi descrito com mais detalhes. O conceito de fator de cristalização é introduzido a fim de gerar uma vizinhança adaptativa para a busca local do recozimento simulado. Em (MARTINS; TSUZUKI, 2009a), heurísticas determinísticas foram incorporadas à proposta original com o objetivo de reduzir o espaço de soluções.

O parâmetro de controle de posição adotado nestes trabalhos admitia apenas o posicionamento ao longo do contorno da região livre de colisão. Uma vez que a busca em parâmetros contínuos é geralmente mais custosa, devido ao espaço infinito de busca, alternativas foram propostas para a definição da posição dos itens. Em (MARTINS; TSUZUKI, 2010), a heurística de posicionamento foi alterada de modo que o posicionamento em vértices da região livre de colisão possuísse maior probabilidade de ocorrência. Os resultados mostraram uma redução no tempo de convergência do algoritmo sem comprometer os resultados. A restrição foi intensificada ao limitar o posicionamento apenas para os vértices da região livre de colisão em (MARTINS; TSUZUKI, 2009b). Os resultados indicaram que a discretização do espaço de posicionamento adotada é vantajosa.

Sato, Martins e Tsuzuki (2013) desenvolveram um algoritmo robusto para realizar operações Booleanas não regularizadas entre polígonos, que foi aplicado para a determinação da região livre de colisão. Até então, eram empregadas operações Booleanas regularizadas, resultando em uma região incompleta, que impossibilitava posicionamentos de encaixe. A busca local foi modificada para priorizar posicionamentos de encaixe, e os testes com problemas tipo “quebra-cabeça” indicaram uma redução do tempo para convergência de 15 vezes em relação aos trabalhos anteriores. Problemas de empacotamento irregular de fita foram resolvidos em (SATO; MARTINS; TSUZUKI, 2012) a partir da adoção de um algoritmo de dois níveis, sendo que o nível externo controlava o comprimento do contêiner. Os resultados obtidos foram bastante competitivos, entretanto, o tempo de execução do algoritmo foi bastante elevado, devido ao tamanho do espaço de soluções.

A priorização das posições de encaixe ocorria do seguinte modo: quando um novo item era escolhido para ser inserido no contêiner, a região livre de colisão era determinada. Se fossem detectadas posições de encaixe, o item era obrigatoriamente inserido nesta posição, caso contrário, era obedecido o parâmetro de posicionamento controlado pelo recozimento simulado. Análises dos resultados mostraram que a ocorrência de posicionamentos de

encaixe era relativamente baixa. Mesmo assim, foi observado um aumento na compactação dos leiautes finais utilizando a estratégia de priorização de encaixes.

Isto posto, como extensão das abordagens anteriores, é proposto neste trabalho um método capaz de obter o posicionamento de encaixe entre dois itens. O objetivo é intensificar posicionamentos que podem contribuir para a compactação, de modo a acelerar a convergência do algoritmo. O algoritmo desenvolvido é testado com os mesmos problemas da literatura a fim de comparar os resultados com os obtidos em trabalhos anteriores.

3.3 Conceitos geométricos

A principal ferramenta geométrica adotada nesta primeira proposta é a região livre de colisão, que combina as informações fornecidas pelo polígono de obstrução e o polígono de posicionamento interno. A região livre de colisão proporciona a obtenção direta das posições de encaixe, que constituem uma importante informação no problema de empacotamento. Nesta seção, o polígono de obstrução e o polígono de posicionamento interno, brevemente introduzidos na [seção 2.2](#), são descritos com maior detalhe. Enfoque é dado a uma definição mais formal do conceito, a fim de facilitar a descrição da metodologia de obtenção automática das posições de encaixe. Ademais, os conceitos de região livre de colisão e posição de encaixe são propriamente introduzidos.

3.3.1 Polígono de obstrução

O polígono de obstrução descreve, para um par de itens, as configurações relativas não válidas dadas às condições geométricas de problemas de empacotamento, i.e., sem sobreposição. Na definição do polígono de obstrução, os itens do par são denominados item móvel e item fixo. Assim, o polígono de obstrução representa as translações que, aplicadas ao item móvel, ocasionam a sobreposição com o item fixo. Desta forma, é possível representar matematicamente o polígono de obstrução como um conjunto de translações. Para um item P_k , considere $i(P_k)$ o seu interior, ∂P_k o seu contorno e $c(P_k)$ seu complemento.

Definição 3.1 *Considere dois polígonos P_i e P_j posicionados no espaço. 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 à sobreposição com P_i . Desta forma,*

$$\Upsilon(P_i, P_j) = \{ \vec{v} \mid \exists \mathbf{a} \in i(P_j), \mathbf{a} + \vec{v} \in i(P_i) \}. \quad (3.5)$$

O polígono de obstrução mapeia o conjunto de translações aplicadas a um item em uma região do espaço, que é definida a partir do ponto de referência do item, como mostra a [Figura 11](#). A partir do exemplo da figura, é possível observar que o polígono de

Figura 11 – Mapeamento de translações fornecendo o polígono de obstrução

Fonte: o autor

obstrução pode ser descrito utilizando apenas as informações da posição do item fixo P_i e do ponto de referência do item móvel P_j . Assim,

$$\Upsilon(P_i + \vec{v}, P_j) = \Upsilon(P_i, P_j) + \vec{v}. \quad (3.6)$$

Devido a esta propriedade, os polígonos de obstrução podem ser determinados em uma etapa anterior à detecção de colisão a partir da definição de uma posição arbitrária para o item fixo. Assim, para detectar a sobreposição de itens, é necessário aplicar uma translação ao polígono de obstrução de acordo com a posição do item fixo no leiaute.

3.3.2 Polígono de posicionamento interno

Para avaliar se um leiaute é válido, duas condições devem ser observadas: não deve haver sobreposição entre os itens e os mesmos devem estar completamente contidos no contêiner. A primeira condição pode ser satisfeita utilizando o polígono de obstrução e, no caso da segunda restrição, pode ser empregado o conceito de polígono de posicionamento interno.

O polígono de posicionamento interno (*inner-fit polygon*) é um conceito derivado do polígono de obstrução e compartilha a mesma ideia de mapear translações para uma região no espaço. No caso do polígono de posicionamento interno, a região mapeada representa todas as translações para um item P_i móvel que o posiciona completamente dentro de um contêiner $\mathcal{C}$. A Figura 12 apresenta um exemplo de polígono de posicionamento interno.

Definição 3.2 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,

$$\Lambda(\mathcal{C}, P_i) = \{\vec{v} \mid \forall \mathbf{a} \in i(P_i), \mathbf{a} + \vec{v} \in \mathcal{C}\}. \quad (3.7)$$

Figura 12 – Polígono de posicionamento interno

Fonte: o autor

O posicionamento de um item P_i em um contêiner utilizando o polígono de posicionamento interno apresenta as seguintes propriedades:

- caso o ponto de referência do item P_i pertencer a $i(\Lambda(\mathcal{C}, P_i))$, P_i é interno a $\mathcal{C}$;
- caso o ponto de referência do item P_i pertencer a $\partial\Lambda(\mathcal{C}, P_i)$, P_i é interno e encosta em $\mathcal{C}$;
- caso o ponto de referência do item P_i pertencer a $c(\Lambda(\mathcal{C}, P_i))$, alguma região de P_i é externa a $\mathcal{C}$.

3.3.2.1 Obtenção do polígono de posicionamento interno

O problema de empacotamento irregular de fita estudado considera apenas contêineres retangulares. Neste caso específico, é simples a obtenção do polígono de posicionamento interno. O polígono será sempre retangular, restando apenas definir as coordenadas dos seus quatro vértices. Estas coordenadas dependem apenas de dois fatores: o ponto de referência do item móvel e as dimensões de seu retângulo envolvente.

Representam-se a altura e largura do retângulo envolvente do item P_i como A_i e L_i e a altura e largura do contêiner como A_c e L_c , respectivamente. Assim, as dimensões, altura e largura, do polígono de posicionamento interno retangular são $A_c - A_i$ e $L_c - L_i$, respectivamente. O ponto inferior esquerdo pode ser definido como o ponto de referência do item P_i quando o item está posicionado na posição inferior esquerda. A partir destas informações é possível definir e posicionar o polígono de posicionamento interno (ver [Figura 13](#)).

Figura 13 – Obtenção do polígono de posicionamento interno

Fonte: o autor

3.3.3 Região livre de colisão

A região livre de colisão é um conceito similar ao espaço de configuração, que é empregado na área de planejamento de trajetória de robôs. Este conceito foi empregado no problema de empacotamento irregular em (MARTINS; TSUZUKI, 2006).

Utilizando apenas os conceitos de polígono de obstrução e polígono de posicionamento interno é possível determinar se um leiaute é válido ou não. Uma estratégia consiste em verificar, para cada item, se o seu ponto de referência está no interior ou no contorno do polígono de posicionamento interno e é externo aos polígonos de obstrução induzidos a ele. A região livre de colisão reúne todas as informações dos polígonos de obstrução e do polígono de posicionamento interno, gerando uma região mapeada que representa, para um determinado item, todas as translações válidas para o seu posicionamento no leiaute. Desta forma, é necessária apenas uma verificação para determinar se o item está em uma posição válida.

Considere um contêiner $\mathcal{C}$ e um conjunto de itens $\mathcal{P} = \{P_1, \dots, P_n\}$ já posicionados. Deseja-se inserir um novo item P_m , $m > n$ no interior do contêiner, sem sobreposição com os itens em $\mathcal{P}$. A região livre de colisão representa o conjunto de translações válidas para o item P_m .

Definição 3.3 A região livre de colisão é o conjunto de todas as translações que, aplicadas a um item específico, o posiciona completamente no interior do contêiner sem que haja colisão com os itens já posicionados.

A Figura 15 mostra um exemplo de região livre de colisão. Quando não existem itens posicionados, a região livre de colisão é o polígono de posicionamento interno. Desta

forma, é possível afirmar que o polígono de posicionamento interno é um caso particular da região livre de colisão.

A região livre de colisão é útil para construção do leiaute utilizando o posicionamento sequencial dos itens, sempre mantendo o leiaute válido. Neste caso, para cada item, a região livre de colisão é calculada e, para realizar o posicionamento, é apenas necessário selecionar algum ponto no interior ou no contorno da região. A Figura 14 apresenta todos os passos para a construção completa de um leiaute utilizando o conceito de região livre de colisão. No posicionamento do quinto item, a região livre de colisão é mapeada para uma aresta apenas. Neste caso, diz-se que a região livre de colisão é degenerada.

Figura 14 – Leiaute válido obtido utilizando a região livre de colisão

Fonte: o autor

3.3.3.1 Obtenção da região livre de colisão

Considere uma situação em que $n - 1$ itens já foram posicionados em um contêiner $\mathcal{C}$. O conjunto destes itens é representado por $\mathcal{P} = \{P_1, \dots, P_{n-1}\}$. Antes de determinar a região livre de colisão, o polígono de posicionamento interno $\Lambda(\mathcal{C}, P_n)$ e todos os polígonos de obstrução induzidos pelos itens fixos $\Upsilon(P_k, P_n)$, onde $k = 1, \dots, n - 1$, devem ser definidos.

Figura 15 – Determinação da região livre de colisão

Fonte: o autor

A região livre de colisão pode ser obtida através da subtração dos polígonos de obstrução induzidos pelos itens fixos do polígono de posicionamento interno do item móvel.

Assim, a região livre de colisão, denotada por $\Pi(\mathcal{C}, \mathcal{P}, P_n)$, pode ser determinada através da expressão:

$$\Pi(\mathcal{C}, \mathcal{P}, P_n) = \Lambda(\mathcal{C}, P_n) - \bigcup_{P_i \in \mathcal{P}} \Upsilon(P_i, P_n). \quad (3.8)$$

A Figura 15 mostra um exemplo no qual três itens já foram posicionados e o polígono de posicionamento interno e os polígonos de obstrução foram definidos.

3.3.4 Posições de encaixe

As posições de encaixe são classificadas neste trabalho como: posições de encaixe deslizante e posições de encaixe perfeito. Ambas são utilizadas nesta proposta para a compactação do leiaute. A região livre de colisão pode ser utilizada para detectar estas posições. Quando existe a possibilidade de encaixe para um item, sua região livre é degenerada, o que significa que esta apresenta arestas e/ou vértices degenerados. O posicionamento em encaixe é de difícil detecção, pois requer alta precisão na obtenção da região livre de colisão e a não detecção destas posições pode ocasionar a perda de mínimos locais.

Figura 16 – Posição de encaixe deslizante

Fonte: o autor

Figura 17 – Posição de encaixe perfeito

Fonte: o autor

Posições de encaixe deslizante correspondem a posicionamentos nos quais o item, depois de inserido no leiaute, pode ser deslocado em um número finito de direções. A Figura 16 mostra um exemplo de encaixe deslizante em que o item quadrado pode apenas se mover horizontalmente. No caso ilustrado, ao posicionar o item móvel no vértice esquerdo da aresta degenerada, o leiaute resultante é o mais compacto. Não é possível obter nenhum leiaute tão compacto sem realizar o encaixe deslizante.

Itens posicionados em encaixe perfeito, uma vez já contidos no leiaute, não podem se movimentar em nenhuma direção. Um exemplo de encaixe perfeito pode ser observado na Figura 17. Neste caso, o posicionamento em encaixe perfeito também constitui a maior compactação dadas todas as configurações possíveis para os dois itens.

3.4 Obtenção das posições de encaixe

Posições de encaixe são verificadas nos leiautes eficientes de problemas de empacotamento irregular. O desafio é encontrar uma heurística construtiva capaz de maximizar o número de posicionamentos de encaixe no leiaute final, uma vez que a análise das soluções parciais geralmente não reflete a qualidade da solução final.

Em uma heurística construtiva, os itens são adicionados no leiaute de forma sequencial. A fim de obter uma maior compactação, um item sempre que possível deve ser inserido em uma posição de encaixe. Quando não existe esta possibilidade, é proposto utilizar um item adicional para criar as condições necessárias a fim de realizar este tipo de posicionamento. Para definir a posição dos dois itens (o próximo item da sequência e o item adicional), é desenvolvido um método capaz de mapear todas as posições para o item adicional que geram a possibilidade de posicionamentos de encaixe para o item atual.

A região livre de colisão $\Pi(\mathcal{C}, \mathcal{P} \cup \{P_n\}, P_{n+1})$ descrita na [Equação 3.8](#) pode ser reescrita como uma operação entre dois polígonos:

$$\Pi(\mathcal{C}, \mathcal{P} \cup \{P_n\}, P_{n+1}) = \Pi(\mathcal{C}, \mathcal{P}, P_{n+1}) - [\vec{v} + \Upsilon(P_n, P_{n+1})] \quad (3.9)$$

Assim, o problema pode ser expresso da seguinte forma: obter as translações para o polígono de obstrução $\Upsilon(P_n, P_{n+1})$ de tal forma que, ao ser subtraído da região livre de colisão $\Pi(\mathcal{C}, \mathcal{P}, P_{n+1})$, ocorra a geração de novas arestas ou vértices degenerados.

3.4.1 Regras para obtenção de aresta e vértices degenerados

Como discutido na [subseção 3.3.4](#), posições de encaixe correspondem a arestas ou vértices degenerados da região livre de colisão. A região livre de colisão é obtida neste trabalho utilizando operações Booleanas não regularizadas que envolvem polígonos de obstrução e o polígono de posicionamento interno, a partir da [Equação 3.8](#). O algoritmo robusto de operações Booleanas não regularizadas adotado é descrito em ([SATO; MARTINS; TSUZUKI, 2013](#)). No trabalho citado, é dada uma atenção especial para as regras de criação de arestas e vértices degenerados, que são expostas de modo breve neste texto.

Em operações Booleanas não regularizadas de subtração, arestas degeneradas são criadas quando há coincidência de arestas com uma mesma orientação. Denotando e_Π como uma aresta da região livre de colisão $\Pi(\mathcal{C}, \mathcal{P}, P_{n+1})$ e e_Υ como uma aresta do polígono de obstrução $\Upsilon(P_n, P_{n+1})$, ambas com a mesma direção e sentido, uma aresta degenerada é criada quando uma translação $\vec{v}$ é aplicada ao polígono de obstrução $\Upsilon(P_n, P_{n+1})$ de modo que:

$$e_\Pi \cap (\vec{v} + e_\Upsilon) \neq \emptyset. \quad (3.10)$$

Como já observado em (SATO; MARTINS; TSUZUKI, 2013), os pontos de intersecção entre dois polígonos de obstrução são posições válidas para o posicionamento, desde que estejam no interior ou contorno do polígono de posicionamento interno. O mesmo é válido para cruzamentos entre polígono de obstrução e o polígono de posicionamento interno. Consequentemente, o conjunto dos pontos de intersecção sempre pertence à região livre de colisão.

Os demais vértices da região livre de colisão são gerados a partir das arestas originais do polígono de obstrução e, por este motivo, sempre são pontos finais de alguma aresta. A partir destas observações, é possível concluir que vértices degenerados só podem ser criados a partir do cruzamento de arestas de dois polígonos de obstrução, uma vez que o caso degenerado não ocorre quando o vértice é ponto inicial ou final de alguma aresta.

A fim de garantir que o vértice é degenerado, nenhuma aresta ligada a ele deve ser incluída na região livre de colisão resultante. Esta condição pode ser expressa em termos de comparação de setores utilizando o conceito de vizinhança definido em (HOFFMANN, 1989). A vizinhança de um ponto p em relação a um polígono A é definida como a intersecção do polígono A com um disco d_ϵ de raio ϵ infinitesimal, centrado no ponto p .

Considere dois vértices, que podem ser acrescentados em qualquer ponto ao longo do contorno de um dos polígonos: um vértice v_Π na região livre de colisão e um vértice v_Υ no polígono de obstrução. Definem-se o setor s_Π , originado de $\Pi(\mathcal{C}, \mathcal{P}, P_{n+1})$ e centrado em v_Π , e o setor s_Υ , obtido a partir de $\Upsilon(P_n, P_{n+1})$ e centrado em v_Υ . Um vértice degenerado é obtido se s_Π contém s_Υ . Desta forma, um vértice degenerado é criado quando uma translação $\vec{v}$ é aplicada ao polígono de obstrução $\Upsilon(P_n, P_{n+1})$ de modo que:

$$\mathcal{F}(s_\Pi) = \vec{v} + \mathcal{F}(s_\Upsilon) \quad (3.11)$$

$$(\vec{v} + s_\Upsilon) \subseteq s_\Pi \quad (3.12)$$

onde $\mathcal{F}(s_\Pi)$ e $\mathcal{F}(s_\Upsilon)$ representam as coordenadas do centro dos setores s_Π e s_Υ , respectivamente. A Figura 18 e a Figura 19 mostram exemplos onde as condições para criação de aresta e vértice degenerados são satisfeitas, respectivamente.

3.4.2 Região degenerada

Considere um conjunto de itens já posicionados em um contêiner retangular $\mathcal{C}$. Deseja-se inserir dois novos itens P_n e P_{n+1} , um a um. O item P_n é posicionado antes de P_{n+1} , de modo que este último seja alocado em uma posição de encaixe deslizante ou de encaixe perfeito, sendo seu movimento restrito obrigatoriamente pelo item P_n e possivelmente por outros itens e/ou paredes do contêiner. Isto implica que a região de colisão do item P_{n+1} é degenerada e o posicionamento do item deve ser realizado em uma

Figura 18 – Obtenção de aresta degenerada

Fonte: o autor

Figura 19 – Obtenção de vértice degenerado

Fonte: o autor

aresta ou vértice degenerado que são gerados devido ao posicionamento do item P_n . Desta forma, busca-se por posicionamentos para o item P_n adequados.

Definição 3.4 Considere um leiaute formado pelo conjunto de itens $\mathcal{P} = \{P_1, \dots, P_{n-1}\}$ em um contêiner $\mathcal{C}$ e dois itens P_n e P_{n+1} não posicionados. A região degenerada induzida pelo segundo item móvel P_{n+1} , o contêiner $\mathcal{C}$ e o conjunto de itens fixos $\mathcal{P}$ ao primeiro item móvel P_n , denotada por $\Psi(\mathcal{C}, \mathcal{P}, P_n, P_{n+1})$, é o conjunto de vetores de translações que, aplicado a P_n , resulta em uma região livre de colisão para o item P_{n+1} que possui aresta(s) degenerada(s) e/ou vértice(s) degenerado(s) não equivalentes aos encontrados na região livre de colisão de P_{n+1} quando desconsiderado o posicionamento do item P_n .

Ou seja, $\Psi(\mathcal{C}, \mathcal{P}, P_n, P_{n+1})$ representa o conjunto de translações $\vec{v}$, tal que:

$$\Pi(\mathcal{C}, \mathcal{P} \cup \{P_n\}, P_{n+1}) = \Pi(\mathcal{C}, \mathcal{P}, P_{n+1}) - [\vec{v} + \Upsilon(P_n, P_{n+1})] \quad (3.13)$$

possui aresta(s) degenerada(s) e/ou vértice(s) degenerado(s) que não são encontrados em $\Pi(\mathcal{C}, \mathcal{P}, P_{n+1})$.

A região degenerada pode ser mapeada para uma região no espaço a partir da definição de um ponto de referência para o item P_n . De modo a simplificar a compreensão e implementação do algoritmo, o ponto de referência adotado é o mesmo definido para o polígono de obstrução.

Para a determinação da região degenerada, é necessária a análise dos pares de arestas e vértices, cada um proveniente de um polígono distinto. Este problema combinatório pode requerer um tempo computacional muito grande para ser processado, especialmente nos casos em que os polígonos possuem muitas arestas. A fim de reduzir o impacto da geometria dos polígonos, é proposta uma implementação mais eficiente para a comparação de arestas e setores.

3.4.3 Determinação de arestas de mesma orientação

A forma mais direta de detectar arestas de igual orientação consiste na comparação de cada par de arestas, sendo que uma é proveniente de $\Pi(\mathcal{C}, \mathcal{P}, P_{n+1})$ e a outra de $\Upsilon(P_n, P_{n+1})$. O algoritmo resultante é combinatório de complexidade $O(m_\Pi \times m_\Upsilon)$, onde m_Π e m_Υ são o número total de arestas de $\Pi(\mathcal{C}, \mathcal{P}, P_{n+1})$ e $\Upsilon(P_n, P_{n+1})$, respectivamente.

Duas informações são necessárias para a determinação da coincidência de arestas de mesma orientação: inclinação e sentido. Estas propriedades são combinadas para gerar uma lista de aresta para cada polígono. Se estas listas são ordenadas em relação à inclinação das arestas, é possível obter um algoritmo de comparação de complexidade $O(m_\Pi + m_\Upsilon)$.

A Figura 20 mostra um exemplo de duas listas de arestas ordenadas em relação à inclinação. A união das duas listas possibilita a obtenção direta das arestas com orientação igual, que, no exemplo ilustrado, consiste nos pares de arestas (p_1, r_1) , (p_2, r_2) , (p_5, r_2) , (p_6, r_3) , e (p_8, r_4) .

Figura 20 – Criação da lista ordenada de arestas

Fonte: o autor

3.4.4 Algoritmo para comparação de setores

Assim como no caso do algoritmo para encontrar todas as arestas com uma mesma orientação, a comparação exaustiva de todos os pares de setores é uma estratégia ineficiente. Na comparação de setores, devem ser considerados os setores originados a partir das combinações das arestas e vértices dos dois polígonos, resultando em uma análise completa de complexidade $O(m_\Pi \times m_\Upsilon)$. A fim de reduzir a complexidade, os setores são descritos utilizando duas listas, ordenadas de modo que seja possível a eliminação de comparações desnecessárias.

Alguns conceitos devem ser introduzidos antes da descrição do algoritmo mais eficiente. A lista $L_{\Pi d}$, denominada lista direta, é definida como uma lista ordenada em relação à inclinação das arestas orientadas de $\Pi(\mathcal{C}, \mathcal{P}, P_{n+1})$, e cada uma é associada a seu vértice de início. Esta lista é similar à empregada na detecção de aresta de mesma orientação. Uma segunda lista $L_{\Pi i}$, denominada lista inversa, armazena todas as arestas de $\Pi(\mathcal{C}, \mathcal{P}, P_{n+1})$, com orientação invertida, e também é ordenada em relação à inclinação das arestas. A obtenção das listas direta e inversa utilizadas para realizar a comparação de setores pode ser observada no exemplo da Figura 21.

Figura 21 – Obtenção das listas direta e indireta

Fonte: o autor

Por causa da informação de vértice de início atribuída a cada aresta, é possível obter o setor centrado no vértice diretamente das duas listas. Este setor é representado por duas arestas que possuem o mesmo vértice de início, uma dessas arestas possui orientação direta e a outra orientação inversa (ver Figura 22). Os setores correspondentes às arestas podem ser obtidos inserindo um vértice no ponto médio de cada aresta. A Figura 23 mostra um exemplo da determinação completa das listas direta e inversa.

Um setor do polígono de obstrução $\Upsilon(P_n, P_{n+1})$ também pode ser representado de maneira similar, utilizando uma aresta $e_{\Upsilon d}$ da lista de arestas diretas $L_{\Upsilon d}$ e uma aresta $e_{\Upsilon i}$ da lista de arestas inversas $L_{\Upsilon i}$. Para cada setor de $\Upsilon(P_n, P_{n+1})$, as posições de inserção das arestas $e_{\Upsilon d}$ e $e_{\Upsilon i}$ na lista $L_{\Pi d}$ são determinadas. A partir destas posições é gerada a lista $E_{\Pi d}$, que contém todas as arestas da lista $L_{\Pi d}$ localizadas entre $e_{\Upsilon d}$ e $e_{\Upsilon i}$. A lista $E_{\Pi d}$ contém todas as arestas internas ao setor s_{Υ} , quando os vértices de início de $e_{\Upsilon d}$ e

Figura 22 – Obtenção do setor a partir das listas direta e indireta

Fonte: o autor

Figura 23 – Determinação completa das listas direta e indireta

Fonte: o autor

e_{γ_i} coincidem com os vértices iniciais destas arestas. O mesmo procedimento pode ser aplicado às arestas de $L_{\Pi i}$. Neste caso, o conjunto $E_{\Pi i}$ é obtido.

Se ambas as arestas de um setor de $\Pi(\mathcal{C}, \mathcal{P}, P_{n+1})$ pertencem às listas $E_{\Pi d}$ e $E_{\Pi i}$, pode-se concluir que as arestas estão contidas pelo setor s_{γ} . Esta é uma condição necessária para determinar se este setor está contido em s_{γ} . Uma vez realizada esta verificação, duas

possibilidades existem para a sequência de arestas no setor. Para verificar a condição suficiente deve-se certificar que a aresta da lista $L_{\Pi i}$ está entre a aresta da lista $L_{\Pi d}$ e $e_{\Upsilon i}$.

A comparação de setores ilustrada na Figura 24 obteve três posicionamentos onde um setor de $\Upsilon(P_n, P_{n+1})$, representado pelo polígono à esquerda na figura, contém um setor de $\Pi(\mathcal{C}, \mathcal{P}, P_{n+1})$, representado na Figura 23. Neste exemplo, é analisado o setor formado pelas arestas p_1 e p_2 . A lista $E_{\Pi i}$ contém todas as arestas da lista inversa $L_{\Upsilon i}$, que consiste no conjunto $\{(v_4, a_{11}), (v_2, a_{12}), (v_2, a_{22}), (v_5, a_{21})\}$. Já a lista $E_{\Pi d}$, que é composta por todas as arestas da lista direta $L_{\Upsilon d}$ contidas pelo setor analisado, é definida pelo conjunto $\{(v_2, a_{21}), (v_5, a_{22}), (v_6, a_{32}), (v_3, a_{31})\}$. É possível observar que duas arestas dos setores v_2 (a_{21} e a_{12}), v_3 (a_{31} e a_{22}) e v_5 (a_{21} e a_{22}) estão contidas nas duas listas, obedecendo à condição necessária. A condição suficiente também é verificada para os três setores obtidos, representados na figura, e, desta forma, estão contidos no setor analisado formado pelas arestas p_1 e p_2 .

Figura 24 – Comparação de um setor do polígono de obstrução com os setores da região livre de colisão da Figura 23

Fonte: o autor

Se a busca binária for utilizada para determinar as posições de inserção, a complexidade do algoritmo é reduzida para $O(m_{\Upsilon} \log m_{\Pi})$.

3.4.5 Construção da região degenerada

Após a aplicação dos algoritmos de comparação, é obtido um conjunto de pares aresta/aresta, aresta/vértice, vértice/aresta ou vértice/vértice que possuem o potencial de gerar posições de encaixe. Estes pares devem ser processados a fim de obter as translações correspondentes para o primeiro item móvel, de acordo com a [Equação 3.13](#). Este processo é realizado separadamente para cada par e a união dos conjuntos resulta na região degenerada utilizada para realizar o posicionamento.

O par aresta/aresta corresponde a uma aresta na região degenerada. Esta aresta pode ser obtida utilizando o método de escorregamento, no qual a aresta móvel e_Υ , que pertence ao polígono de obstrução $\Upsilon(P_n, P_{n+1})$, desliza ao longo da aresta fixa e_Π , que compõe a região livre de colisão $\Pi(\mathcal{C}, \mathcal{P}, P_{n+1})$. A partir de uma configuração na qual o par se encosta, a aresta móvel e_Υ desliza na direção definida por e_Π até que as arestas se coincidam em apenas um ponto. A região degenerada correspondente é obtida a partir do caminho percorrido pelo ponto de referência do polígono de obstrução. A [Figura 25](#) mostra um exemplo deste método.

Figura 25 – Determinação de uma aresta da região degenerada através do método de escorregamento

Fonte: o autor

O método de escorregamento também pode ser utilizado para obter as demais translações que compõem a região degenerada. Todavia, os últimos dois casos são mais simples e a obtenção pode ser feita de modo direto. Os pares mistos (vértice/aresta ou aresta/vértice) obtidos em uma comparação de setor correspondem às arestas da região degenerada. Uma aresta da região degenerada originada de um par misto é obtida a partir de uma translação correspondente à distância do vértice ao ponto de referência. O mesmo procedimento pode ser aplicado para processar um par de vértices obtido a partir do algoritmo de comparação de setores.

3.4.6 Restrição do posicionamento

A região degenerada pode ser definida a partir das condições de criação de arestas degeneradas e vértices degenerados, utilizando os algoritmos de determinação de arestas de mesma orientação e da comparação de setores. A região degenerada representa translações para o primeiro item móvel, P_n . Como este item fará parte do leiaute, o seu posicionamento deverá ser válido, o que significa que o item deve estar completamente contido no contêiner. Para realizar o posicionamento válido, a translação aplicada deve estar representada na região livre de colisão do item, $\Pi(\mathcal{C}, \mathcal{P}, P_n)$. Desta forma, o posicionamento final deve ser utilizar a combinação da região degenerada com a região livre de colisão, que resulta na região degenerada restrita.

3.4.7 Estudo de caso

O posicionamento de dois itens no leiaute é realizado através do conceito da região degenerada restrita. Dado um leiaute incompleto, em que ainda faltam posicionar itens, e o par P_n e P_{n+1} que será inserido, a obtenção desta ferramenta geométrica é realizada em 6 etapas:

1. determinar o polígono de obstrução induzido pelo item fixo P_{n+1} ao item móvel P_n ;
2. determinar a região livre de colisão do item P_{n+1} ;
3. detectar todas as arestas coincidentes de mesma orientação entre o polígono de obstrução e a região livre de colisão utilizando o algoritmo descrito na [subseção 3.4.3](#);
4. detectar todos os pares que contém aresta e/ou vértice a partir do algoritmo descrito na [subseção 3.4.4](#);
5. para cada par de elementos obtido no [Item 3](#) e no [Item 4](#), acrescentar a região degenerada utilizando o processo apresentado na [subseção 3.4.5](#);
6. restringir a região degenerada utilizando o método descrito na [subseção 3.4.6](#).

A [Figura 26](#) apresenta os dados de entrada para o exemplo em questão. O leiaute é considerado vazio, e dois itens de geometria simples são considerados. As duas primeiras etapas resultam na determinação do polígono de obstrução (vide [Figura 27](#)) e da região livre de colisão (vide [Figura 28](#)). Em seguida, é realizada a busca por arestas com direção e sentido coincidentes, cujo resultado é mostrado na [Figura 29](#). Antes da geração da região degenerada, deve-se processar a comparação de setores, que obtém alguns pares de arestas, vértices e alguns pares mistos, que são mostrados na [Figura 30](#). A partir destes pares de elementos, a região degenerada é construída, como mostrada na [Figura 31](#). Por fim, o

resultado final desejado é obtido realizando a restrição de posicionamentos que ocasionam extrapolação dos itens, que pode ser observado na Figura 32. A Figura 33 exibe alguns exemplos de posicionamento na região degenerada e demonstra que o outro item pode ser inserido em uma posição de encaixe.

Figura 26 – Itens e contêiner para o caso exemplo

Fonte: o autor

Figura 27 – Polígono de obstrução para o caso exemplo

Fonte: o autor

Figura 28 – Região livre de colisão para o caso exemplo

Fonte: o autor

Figura 29 – Comparação de arestas para o caso exemplo

Fonte: o autor

Figura 30 – Comparação de setores para o caso exemplo

Fonte: o autor

Figura 31 – Região degenerada para o caso exemplo

Fonte: o autor

Figura 32 – Restrição da região degenerada para o caso exemplo

Fonte: o autor

3.5 Aplicação no problema de empacotamento irregular de fita

O método que utiliza a região degenerada considera apenas posicionamentos válidos e, conseqüentemente, a sobreposição entre itens não é permitida. Desta forma, deve ser realizada uma busca na sequência, de modo que a definição adequada das estratégias de obtenção das posições e orientações é necessária para obter uma solução eficiente.

Isto posto, é possível definir dois subproblemas: a escolha do método de busca e a

Figura 33 – Exemplos de posicionamento para o caso exemplo

Fonte: o autor

definição da heurística de posicionamento, que deve considerar a possibilidade de múltiplas orientações. Além disso, existe a questão da definição da função objetivo. Uma vez que se deseja maximizar a ocorrência de posições de encaixe, é cogitada a possibilidade da adoção de um contêiner de dimensões fixas. Neste caso, a função objetivo consiste na área desperdiçada do contêiner e a compactação do leiaute é obtida a partir da redução gradual do comprimento. Como nenhuma sobreposição não é permitida, soluções geradas em contêiner fixo podem não conter todos os itens.

3.5.1 Heurísticas de posicionamento

O primeiro requisito da heurística de posicionamento é o posicionamento simultâneo de dois itens, como previsto na concepção da região degenerada. Outra restrição adotada é considerar apenas vértices ou pontos de intersecção da região degenerada no posicionamento, a fim de reduzir o espaço de busca.

Entretanto, há casos em que não é possível realizar o posicionamento utilizando o par de itens. Isso ocorre quando não há espaço livre suficiente para posicionar ambos os itens ou quando existe um número ímpar de itens. Nestes casos, os itens são posicionados utilizando a heurística inferior esquerda, com o auxílio da região livre de colisão, que possui a capacidade de preencher lacunas no leiaute.

Uma regra adicional também é considerada para todas as heurísticas: o posicionamento de encaixe tem precedência sobre o posicionamento de par utilizando a região degenerada. O posicionamento de pares emprega dois itens para garantir apenas uma posição de encaixe. Desta forma, esta regra assegura um maior número de itens em situação de encaixe no leiaute final, o que é um dos objetivos principais desta proposta.

Considerando as regras definidas, foram propostas três heurísticas de posicionamento determinísticas: inferior esquerda, forçada e exploração de par, que são descritas a seguir.

- **Inferior esquerda:** são investigadas todas as combinações de posicionamento para os dois itens. A posição escolhida é aquela que minimiza a coordenada na direção do comprimento do centro de gravidade do retângulo envolvente dos dois itens. Em caso de empate, é escolhido o par cujo centro de gravidade possui menor coordenada na direção da largura.
- **Forçada:** antes do posicionamento do par, é realizada uma busca na sequência por itens que podem ser inseridos no leiaute em posição de encaixe perfeito ou deslizante. Se mais de um item é encontrado, a posição do seu centro de gravidade é analisada. É escolhido o item cujo centro de gravidade se localiza mais à esquerda. No caso de empate, é selecionado o item com o menor valor de coordenada vertical do centro de gravidade. Se nenhum item pode ser posicionado em encaixe, a heurística inferior esquerda é utilizada.
- **Exploração de pares:** apenas o primeiro item do par é obtido a partir da sequência. O segundo item selecionado resulta da investigação nos demais itens a serem posicionados. O par e sua posição são escolhidos obedecendo aos critérios da heurística inferior esquerda, i.e., priorizando pares de itens capazes de obter a posição mais à esquerda do centro de gravidade do retângulo envolvente. Caso haja mais de uma configuração com o menor valor de posição horizontal do centro de gravidade, a menor posição vertical é escolhida.

As heurísticas forçada e exploração de par podem ser utilizadas em conjunto, gerando quatro possibilidades para o posicionamento. As quatro variações propostas são: inferior esquerda, forçada, exploração de par e exploração de pares forçada. A [Figura 34](#) mostra um exemplo que ilustra o comportamento básico de cada heurística de posicionamento proposta.

3.5.2 Compactação do leiaute

Soluções de busca no leiaute, que admitem sobreposição durante a busca, muitas vezes utilizam o contêiner de dimensões fixas. A finalidade é a simplificação da função objetivo, que considera apenas a eliminação da sobreposição. O contêiner é reduzido cada vez que um leiaute válido é obtido.

No caso da proposta deste trabalho, a sobreposição não é admitida e, por este motivo, a função objetivo visa apenas à compactação do leiaute. A utilização desta abordagem tem em vista a intensificação da ocorrência de posições de encaixe. Esta estratégia foi aplicada a uma solução construtiva em ([SATO; MARTINS; TSUZUKI, 2012](#)) e obteve bons resultados. O comprimento inicial do contêiner é obtido a partir da execução de uma iteração do algoritmo de construção do leiaute. Em seguida, o contêiner é reduzido,

Figura 34 – Heurísticas de posicionamento de par com priorização de encaixe

Fonte: o autor

possivelmente causando a extrapolação de alguns itens para fora do contêiner. Estes itens são removidos do leiaute e a busca é reiniciada. Cada vez que o problema é resolvido, ou seja, um leiaute válido com todos os itens é gerado, o comprimento do contêiner é reduzido. Caso não seja possível convergir para uma solução completa ao atingir alguma condição de parada, o contêiner é expandido. O procedimento é repetido até que algum critério global de parada é ativado. A [Figura 35](#) mostra um exemplo deste procedimento.

Figura 35 – Compactação do leiaute sem admissão de sobreposição

Fonte: o autor

É necessário definir dois parâmetros, um relativo à redução e outro à expansão do comprimento do contêiner. Cada um dos parâmetros consiste em valores que representam percentualmente a variação do comprimento. Um elevado valor para o parâmetro de

redução pode acelerar o algoritmo, porém o algoritmo pode não ser capaz de obter um leiaute válido no caso de contêiner muito reduzido. Assim, o valor destes parâmetros influencia bastante no desempenho e eficiência do algoritmo.

Ao considerar o contêiner fixo, é possível observar que o problema de empacotamento irregular de fita é convertido em um problema da mochila com apenas um objeto grande (ver [seção 3.1](#)). Usualmente, este problema deve ser resolvido seguidas vezes para obter uma solução satisfatória. Desta forma, apesar do potencial de obtenção de um maior número de posições de encaixe, o desempenho do algoritmo pode ser comprometido quando o contêiner de dimensões fixas é adotado. Por este motivo, as duas alternativas, contêiner fixo ou de dimensão aberta, são consideradas.

3.5.3 Heurística de busca

Assim como em ([SATO; MARTINS; TSUZUKI, 2012](#)), o recozimento simulado é adotado para maximizar a compactação do leiaute. A fim de obter uma boa solução, uma busca local é realizada, que iterativamente aplica pequenas modificações na solução. As modificações consideradas neste trabalho são: a troca de dois itens na sequência e a atribuição de uma nova orientação para um item. O recozimento simulado é utilizado para determinar quais soluções geradas são aceitas e quais são rejeitadas.

O recozimento simulado é uma meta-heurística probabilística que foi originalmente proposta por [Kirkpatrick, Gellat e Vecchi \(1983\)](#) e, nesta abordagem, é aplicado para controlar os parâmetros de rotação de cada item, assim como a sequência de posicionamento. A condição local de convergência é atingida quando um número predeterminado de soluções é aceito. Para uma determinada temperatura do recozimento simulado, a condição de convergência global é acionada quando todas as soluções geradas possuem o mesmo valor da função objetivo. A temperatura apresenta decaimento geométrico, e o parâmetro que controla a redução de temperatura é definido manualmente.

O [Algoritmo 1](#) detalha todas as etapas da solução adotada para o problema de empacotamento irregular de fita. Quatro variações deste algoritmo podem ser obtidas utilizando as diferentes heurísticas de posicionamento propostas na [subseção 3.5.1](#). O parâmetro α controla o decaimento geométrico da temperatura. Quanto menos próximo de 1 for o valor, mais rápido o algoritmo convergirá, no entanto soluções melhores podem não ser encontradas. A função objetivo $F(x)$ varia de acordo com o número de níveis adotado. No caso do algoritmo de um nível, a função objetivo é o comprimento do leiaute. Já no caso da utilização de dois níveis, seu valor é proporcional ao espaço desperdiçado no contêiner.

Algoritmo 1 EMPACOTAENCAIXEPERFEITO

```

 $L \leftarrow \langle \text{Comprimento inicial} \rangle$ 
 $x \leftarrow \langle \text{Solução inicial aleatória} \rangle$ 
 $T \leftarrow \langle \text{Temperatura inicial} \rangle$ 
Enquanto  $\langle \text{Condição global de parada não satisfeita} \rangle$  faça
   $T \leftarrow T * \alpha$ 
  Enquanto  $\langle \text{Condição local de parada não satisfeita} \rangle$  faça
     $val \leftarrow random(0, 1)$ 
    Se  $val < 0.5$  então
       $\mathbf{x}^* \leftarrow \langle \text{Modificar sequência de posicionamento} \rangle$ 
    Senão
       $\langle \text{Seleciona um item} \rangle$ 
       $\mathbf{x}^* \leftarrow \langle \text{Aplica nova orientação ao item selecionado} \rangle$ 
    Fim Se
     $\mathbf{x}^* \leftarrow \langle \text{Heurística de posicionamento com encaixe perfeito} \rangle$ 
     $\Delta E = F(\mathbf{x}^*) - F(\mathbf{x})$ 
    Se  $\Delta E < 0$  então
       $\mathbf{x} \leftarrow \mathbf{x}^*$ 
    Senão
      Se  $random(0, 1) < e^{-\Delta E/kT}$  então
         $\mathbf{x} \leftarrow \mathbf{x}^*$ 
      Fim Se
    Fim Se
  Se  $\langle \text{Algoritmo de um nível} \rangle$  então
     $\langle \text{Encerrar algoritmo} \rangle$ 
  Senão
    Se  $\langle \text{Todos os itens foram posicionados} \rangle$  então
       $L \leftarrow \langle \text{comprimento do leiaute} \rangle$ 
       $\langle \text{Comprimento do contêiner} \rangle \leftarrow L$ 
    Fim Se
  Fim Se
Fim Enquanto
Fim Enquanto

```

3.6 Resultados

O algoritmo de posicionamento de par em encaixe perfeito foi testado com 13 problemas de referência da literatura. A [Tabela 1](#) lista as características de cada problema, como o número e variedade de itens e orientações admitidas. Os dados completos de cada problema podem ser obtidos no site do ESICUP¹. O conjunto de problemas de referência inclui uma grande diversidade, alguns artificiais, incluindo problemas tipo “quebra-cabeça”, cujo ótimo global é conhecido.

Foram desenvolvidas e testadas seis variações do algoritmo: inferior esquerdo (IE), forçado (F), exploração de par (EP), exploração de par forçado (EPF), dimensões fixas

¹ <http://paginas.fe.up.pt/~esicup/tiki-index.php>

Tabela 1 – Dados dos problemas de referência da literatura

Caso	Núm. total de itens	Núm. itens distintos	Núm. médio de vértices	Orientações
Albano	24	8	7,25	$0^\circ, 180^\circ$
Dagli	30	10	6,30	$0^\circ, 180^\circ$
Dighe1	16	16	3,87	0°
Dighe2	10	10	4,70	0°
Fu	12	12	3,58	$0^\circ, 90^\circ, 180^\circ, 270^\circ$
Jakobs1	25	25	5,60	$0^\circ, 90^\circ, 180^\circ, 270^\circ$
Jakobs2	25	25	5,36	$0^\circ, 90^\circ, 180^\circ, 270^\circ$
Mao	20	9	9,22	$0^\circ, 90^\circ, 180^\circ, 270^\circ$
Marques	24	8	7,37	$0^\circ, 90^\circ, 180^\circ, 270^\circ$
Shapes0	43	4	8,75	0°
Shapes1	43	4	8,75	$0^\circ, 180^\circ$
Shapes2	28	7	6,29	$0^\circ, 180^\circ$
Shirts	99	8	6,63	$0^\circ, 180^\circ$
Swim	48	10	21,90	$0^\circ, 180^\circ$
Trousers	64	17	5,06	$0^\circ, 180^\circ$

Fonte: [Gomes e Oliveira \(2006\)](#)

(DF), dimensões fixas de posição livre (DFPL). Os quatro primeiros casos, IE, F, EP e EPF, utilizam o algoritmo que considera o contêiner com comprimento variável, também denominado de algoritmo de um nível. A diferença entre cada método é a heurística de posicionamento adotada. Já os casos DF e DFPL utilizam o contêiner de dimensões fixas e aplicam a estratégia de redução incremental do comprimento, denominado de algoritmo de dois níveis. No caso DF, a heurística de posicionamento é a inferior esquerda, enquanto a variação DFPL utiliza o recozimento simulado para controlar o posicionamento dos itens.

Todos os testes foram executados em um computador Xeon 2.40Ghz com 48GB de memória. Para cada problema estudado, dez execuções do algoritmo foram realizadas. Os seguintes parâmetros foram utilizados:

- para o recozimento simulado:
  - fator de decrescimento da temperatura: 0,9;
  - número máximo de soluções aceitas para cada temperatura: 1000;
- para o algoritmo de dois níveis:
  - fator de redução do comprimento: 1%;
  - fator de expansão do comprimento: 0,3%.

A temperatura inicial do algoritmo de recozimento simulado foi determinada realizando uma breve análise. Várias temperaturas foram testadas e a porcentagem de soluções rejeitadas foi medida durante um laço de temperatura. O valor escolhido foi aquele em que a taxa de rejeição mais se aproximou de 50%. Outro parâmetro definido foi o

tempo limite para a execução dos algoritmos de dois níveis. Nos três casos mais complexos, Shirts, Swim e Trousers, o algoritmo foi limitado a 6 horas segundos de execução. Nos demais casos, o valor adotado foi de 2 horas segundos.

A [Tabela 2](#) e a [Tabela 3](#) apresentam os resultados obtidos para cada variação testada do algoritmo de um e dois níveis, respectivamente. Em alguns testes dos casos Shirts e Swim, o tempo de execução excedeu 28 horas segundos e, por este motivo, os testes se tornaram inviáveis e os resultados foram descartados. Os resultados estão divididos em dois grupos: um nível e dois níveis. Para cada caso da literatura, os valores destacados em negrito são os melhores de cada grupo. É possível observar que, para os resultados dos algoritmos de dois níveis, a variação DF é mais eficiente que DFPL em todos os casos com exceção de Dighe2. No caso de solução com um nível, as heurísticas IE e F obtiveram os melhores resultados. Ademais, o caso IE foi capaz de convergir mais rapidamente que as demais heurísticas.

Tabela 2 – Resultados do algoritmo baseado em posições de encaixe (um nível)

Caso	Densidade média (%)				Tempo de execução médio (s)			
	IE	F	EP	EPF	IE	F	EP	EPF
Albano	84,10	82,27	83,54	81,65	2.152	4.081	6.166	2.565
Dagli	81,24	80,36	80,57	79,09	1.384	4.276	6.703	9.131
Dighe1	79,13	100,00	77,57	100,00	241	135	1.453	486
Dighe2	83,79	88,01	76,63	77,58	138	127	532	597
Fu	87,16	86,36	86,55	86,36	342	747	741	673
Jakobs1	80,99	81,67	80,37	81,67	486	1.542	1.892	2.232
Jakobs2	73,52	74,02	74,30	74,01	806	3.579	1.643	3.531
Mao	78,62	75,94	77,49	75,95	4.449	12.712	10.660	16.076
Marques	84,29	84,57	84,55	83,94	2.021	7.190	4.172	6.645
Shapes0	63,89	64,25	62,79	62,34	1.812	1.218	8.086	4.787
Shapes1	66,50	65,09	65,95	64,59	3.069	5.099	7.269	8.288
Shapes2	77,97	79,38	77,76	78,75	864	1.908	2.440	2.245
Shirts	82,41	–	81,58	–	27.767	–	69.210	–
Swim	64,47	–	–	–	94.568	–	–	–
Trousers	83,80	80,07	83,22	79,11	14.282	77.781	42.797	74.304

Fonte: o autor

A comparação geral entre os métodos de um e de dois níveis pode ser realizada a partir dos dados da [Tabela 4](#). A fim de facilitar essa análise, a execução do algoritmo de dois níveis foi interrompida quando este atingiu o tempo limite definido pela convergência do algoritmo de um nível. Desta forma, ambos os algoritmos consumiram o mesmo período de tempo para obter o leiaute final. Para os casos Shirts e Swim, cuja execução do algoritmo de um nível demandou um tempo demasiadamente alto, foi adotado o tempo limite de 21.600 segundos para a execução. Apenas os melhores resultados, considerando todas as heurísticas, são apresentados. Os valores indicam que o desempenho da variação DF,

Tabela 3 – Resultados do algoritmo baseado em posições de encaixe (dois níveis)

Caso	Densidade média (%)		Tempo de execução (s)
	Inferior esquerda (DF)	Posicionamento livre (DFPL)	
Albano	86,05	84,88	7.200
Dagli	83,89	81,36	7.200
Dighe1	99,96	99,95	7.200
Dighe2	89,54	89,57	7.200
Fu	91,11	89,82	7.200
Jakobs1	88,82	84,46	7.200
Jakobs2	80,10	76,58	7.200
Mao	79,80	77,60	7.200
Marques	87,71	86,54	7.200
Shapes0	65,71	65,07	7.200
Shapes1	69,46	64,06	7.200
Shapes2	80,45	77,38	7.200
Shirts	83,13	79,42	21.600
Swim	65,32	62,55	21.600
Trousers	86,07	79,67	21.600

Fonte: o autor

de dois níveis, é a mais eficiente das soluções, pois obteve os leiautes mais densos em aproximadamente 87% dos casos.

Tabela 4 – Comparação dos resultados do algoritmo baseado em posições de encaixe

Caso	Um nível		Dois níveis		Tempo (s)
	Densidade (%)	Heurística	Densidade (%)	Heurística	
Albano	84,10	IE	85,11	DF	2.152
Dagli	81,24	IE	82,08	DF	1.384
Dighe1	100,00	F	88,41	DFPL	135
Dighe2	88,01	F	88,98	DF	127
Fu	87,16	IE	88,88	DF	342
Jakobs1	81,67	F	87,20	DF	1.542
Jakobs2	74,30	EP	78,40	DF	1.643
Mao	78,62	IE	79,54	DF	4.449
Marques	84,57	F	87,71	DF	7.190
Shapes0	64,25	F	64,29	DF	1.218
Shapes1	66,50	IE	68,94	DF	3.069
Shapes2	79,38	F	78,99	DF	1.908
Shirts	82,41	IE	83,13	DF	21.600
Swim	64,47	IE	65,32	DF	21.600
Trousers	83,80	IE	85,75	DF	14.282

Fonte: o autor

A [Tabela 5](#) mostra uma comparação com as densidades médias dos leiautes obtidos em um trabalho anterior, que propôs o algoritmo de posicionamento exato utilizando a região livre de colisão (PERLC) ([SATO; MARTINS; TSUZUKI, 2012](#)), e com as obtidas pelo algoritmo Guided Cuckoo Search (GCS) ([ELKERAN, 2013](#)), que estão entre as melhores compactações médias da literatura. Em relação ao algoritmo PERLC, os seguintes

resultados foram melhores: Fu, Jakobs1 e Shapes0, porém os tempos de execução para os dois primeiros casos foram maiores. Leiautes com compactação similar, com diferença menor que 5%, foram obtidos nos casos Albano, Dighe1, Jakobs2, Marques e Shapes2. Para os demais problemas, os resultados foram competitivos com exceção dos casos Shirts e Swim, cuja compactação alcançada com o algoritmo PERLC foi maior. No entanto, a variação do tempo de execução foi menor para o algoritmo proposto. Os melhores resultados da literatura são consideravelmente mais eficientes e estão dispostos na tabela como referência.

Tabela 5 – Comparação entre as densidades médias obtidas pelo algoritmo baseado em posições de encaixe e por outras soluções

Caso	Algoritmo proposto		PERLC		GCS	
	Densidade (%)	Tempo (s)	Densidade (%)	Tempo (s)	Densidade (%)	Tempo (s)
Albano	86,05	7.200	86,31	21.600	87,47	1.200
Dagli	83,89	7.200	87,12	21.600	87,06	1.200
Dighe1	99,96	7.200	100,00	600	100,00	600
Dighe2	89,54	7.200	100,00	600	100,00	600
Fu	91,11	7.200	89,54	600	90,93	600
Jakobs1	88,82	7.200	88,61	1.800	88,90	600
Jakobs2	80,10	7.200	80,18	5.400	81,14	600
Mao	79,80	7.200	81,14	21.600	82,93	1.200
Marques	87,71	7.200	87,71	5.400	89,40	1.200
Shapes0	65,71	7.200	65,31	21.600	67,26	1.200
Shapes1	69,46	7.200	71,71	908.175	74,24	1.212
Shapes2	80,45	7.200	80,81	5.400	82,55	1.205
Shirts	83,13	21.600	87,18	1.812.654	88,12	1.200
Swim	65,32	21.600	70,82	5.993.945	74,62	1.200
Trousers	86,07	21.600	87,73	86.400	88,69	1.200

Fonte: o autor

As melhores densidades obtidas pelo algoritmo proposto para cada caso são apresentadas na Tabela 6 em conjunto com as melhores soluções do algoritmo Beam Search (BEN-[NELL](#); [SONG, 2010](#)) e do GCS, que são as melhores da literatura. Os resultados deste trabalho não apresentaram nenhum avanço em relação às soluções geradas pelo método GCS. Já no caso do algoritmo Beam Search, que também utiliza a abordagem de busca na sequência, 4 resultados melhores foram alcançados. A [Figura 36](#) exibe os melhores leiautes obtidos a partir do posicionamento de pares em posições de encaixes.

Tabela 6 – Melhores soluções obtidas em comparação com as melhores na literatura

Caso	Algoritmo proposto			PERLC		Beam Search		GCS	
	D.(%)	Var.	Tempo (s)	D.(%)	Tempo (s)	D.(%)	Tempo (s)	D.(%)	Tempo (s)
Albano	86.66	DF	7.200	89,21	190.342	87,88	5.460	89,58	1.200
Dagli	85.23	DF	7.200	88,36	629.047	87,97	17.331	89,51	1.200
Dighe1	100.00	DF	7.200	100,00	1.704	100,00	1.4	100,00	600
Dighe2	89.61	DF	7.200	100,00	1.229	100,00	0.3	100,00	600
Fu	91.84	DF	7.200	91,96	32.497	90,28	1.192	92,41	600
Jakobs1	89.07	DF	7.200	89,07	7.497	85,96	2.193	89,09 ¹	600
Jakobs2	80.42	DF	7.200	84,83	79.496	80,40	75	87,73	600
Mao	81.90	DF	7.200	84,23	3,195.538	84,07	16.757	85,44	1.200
Marques	88.91	DF	7.200	90,01	74.614	88,92	10.692	90,59	1.200
Shapes0	66.50	DF	7.200	67,59	181.204	64,35	8	68,79	1.200
Shapes1	70.82	DF	7.200	72,52	491.459	72,55	398	76,73	1.200
Shapes2	82.03	DF	7.200	83,30	261.004	81,29	5.603	84,84	1.200
Shirts	84.48	DF	21.600	87,59	2,528.972	89,69	6.217	88,96	1.200
Swim	67.65	IE	21.600	71,78	5,287.061	75,04	15.721	75,94	1.200
Trousers	87.57	DF	21.600	90,07	1,016.331	90,38	5.988	91,00	1.200

¹ Em [Elkeran \(2013\)](#), o autor apresenta 89,10% como valor de compactação máxima para Jakobs2, porém é possível inferir a partir do leiaute que o valor real é 89,09%.

Fonte: o autor

Figura 36 – Leiautes mais compactos obtidos com o algoritmo baseado em posições de encaixe

Fonte: o autor

4 Abordagem rasterizada para o problema de minimização de sobreposição

O algoritmo proposto para a solução de problemas de empacotamento irregular de fita, baseado em posicionamentos de encaixe, foi capaz de obter alguns resultados competitivos, em tempos menores e mais consistentes, de modo a ampliar sua aplicabilidade. No entanto, em comparação com os métodos da literatura, o tempo de execução do algoritmo é bastante alto e a compactação obtida é, no geral, menor.

Um dos problemas identificados foi a dificuldade de obtenção de soluções válidas completas, no caso da estratégia de dois níveis. O algoritmo de recozimento simulado apresentou um elevado número de iterações para convergir para uma solução muitas vezes incompleta. A partir destas observações, foi concluído que o espaço de soluções, do modo como foi definido, prejudicava a busca pela melhor solução. Uma hipótese levantada para explicar a baixa efetividade da busca foi a dificuldade em definir uma boa vizinhança para o problema. Mesmo na variante da solução que permitia o posicionamento livre, isto é, em qualquer vértice da região degenerada, uma mudança na posição de um item alterava diretamente a configuração do par que foi posicionado. Esta perturbação era propagada para o restante do leiaute, gerando soluções vizinhas bastante distintas.

Por estes motivos, é proposta neste trabalho uma nova alternativa para solucionar o problema de empacotamento irregular. Com o intuito de evitar a geração de um espaço de busca desconexo, a estratégia de busca no leiaute foi escolhida. Neste tipo de abordagem, a restrição de sobreposição é relaxada e, conseqüentemente, durante a execução do algoritmo, é permitida a sobreposição entre itens. Na literatura recente, vários autores propuseram soluções de busca no leiaute que empregavam o contêiner de dimensões fixas, visando evitar a utilização de uma função objetiva composta. Estes métodos são chamados de soluções de minimização de sobreposição. Isto posto, a segunda proposta neste trabalho, descrita neste capítulo, consiste em uma solução de minimização de sobreposição para problemas de empacotamento irregular de fita utilizando um novo conceito de ferramenta rasterizada para detectar sobreposição entre itens.

4.1 Discussão sobre a questão da rasterização

A rasterização de geometrias complexas é usualmente empregada em problemas geométricos para simplificar cálculos. Métodos rasterizados realizam um balanço entre precisão, armazenamento de informações (memória) e desempenho. Em linhas gerais, a precisão, que pode ser controlada, determina os requisitos de memória e o desempenho do

algoritmo, de modo que um aumento de precisão é obtido ao custo de um maior consumo de memória e um maior tempo para realizar os cálculos.

Na esfera de problemas de empacotamento, em algumas situações, como no caso de formas que envolvem curvas mais complexas, não existe uma solução simples para lidar com a geometria de forma exata e, por este motivo, é adequada a adoção de algum método aproximado, que pode ser a rasterização da geometria dos itens. No entanto, a maior parte das propostas na literatura para problemas de empacotamento irregular de fita utiliza a representação dos itens por polígonos aproximados. Nestes casos, a ferramenta geométrica mais utilizada é o polígono de obstrução, que combina as propriedades de alta precisão e bom desempenho. Por esta perspectiva, é difícil justificar a utilização do método rasterizado para a detecção de sobreposição de polígonos. A obtenção robusta do polígono de obstrução para itens não convexos é uma tarefa complexa, porém já existem soluções na literatura para este problema.

Soluções do grupo de minimização de sobreposição geralmente apresentam uma estrutura básica comum para resolver o problema do empacotamento. As características principais são: a realização da busca no leiaute, a atribuição de um valor para a sobreposição e a fixação das dimensões do contêiner. A compactação é realizada a partir de reduções incrementais do comprimento do contêiner que, a cada iteração, cria um novo problema de empacotamento de mochila. Este problema é resolvido a partir da definição de uma função sobreposição para relaxar a restrição, a fim de gerar um espaço de busca mais conexo. Uma busca local geralmente é aplicada para eliminar a sobreposição, utilizando várias iterações para resolver o problema.

No caso específico de propostas de minimização de sobreposição, as informações fornecidas pelo polígono de obstrução não são suficientes. Apesar da possibilidade da realização de parte dos cálculos em uma etapa de pré-processamento, ainda é necessário realizar cálculos geométricos relativamente complexos durante a solução do problema. Somando-se a isso a vizinhança de busca infinita e a grande repetição dos cálculos, muitas soluções recorrem a estratégias que limitam o espaço de busca. Em (EGEBLAD; NIELSEN; ODGAARD, 2007) e (UMETANI et al., 2009), os itens são movimentados em apenas uma direção por vez. Já em (IMAMICHI; YAGIURA; NAGAMUCHI, 2009) e (LEUNG; LIN; ZHANG, 2012), é utilizado um modelo de programação não linear que considera as posições relativas e as orientações fixas. Uma exceção pode ser encontrada no método de (ELKERAN, 2013), cuja busca pela posição na vizinhança é realizada através da meta-heurística busca cuco, que não garante a melhor solução dadas as restrições de tempo.

Na proposta deste trabalho, a vizinhança é simplificada a partir da discretização das posições dos itens, obtidas utilizando uma grade. Esta estratégia já foi empregada anteriormente, porém, no conhecimento dos autores, não foi utilizada junto com a aborda-

gem de minimização de sobreposição. É proposta uma inédita rasterização do polígono de obstrução, que garanta um maior controle da precisão na detecção de sobreposição. Isto possibilita a utilização de ferramentas rasterizadas para simplificar e, conseqüentemente, aumentar o desempenho do algoritmo. Além disso, são utilizadas duas estratégias para acelerar os cálculos: o uso de duas resoluções e o processamento paralelo massivo em placas gráficas.

4.2 Problema de minimização de sobreposição

A estratégia de minimização de sobreposição visa relaxar a restrição de sobreposição, transformando o problema original em uma série de subproblemas em que o contêiner possui dimensões fixas. Este subproblema pode ser considerado um problema da mochila, porém, neste trabalho, é referido como problema da minimização de sobreposição. Esta terminologia foi escolhida a fim de diferenciar soluções puras de empacotamento de mochila de soluções utilizadas para resolver problemas intermediários de empacotamento de fita. É importante notar também que, no caso do problema de minimização de sobreposição, é sempre possível obter um leiaute válido utilizando todos os itens, o que normalmente não ocorre em problemas de mochila.

No problema de minimização da sobreposição, a relaxação da condição de não sobreposição é obtida utilizando a função sobreposição. O objetivo é obter uma função que represente, para um par de itens em sobreposição, quão distante a configuração está de ser separada, i.e., sair do estado de sobreposição. Quando os itens já estão em uma disposição válida, o valor da função sobreposição deve ser zero.

O problema pode ser enunciado como: dado um contêiner retangular com dimensões fixas, um conjunto de itens (polígonos quaisquer, podendo ser não convexos) e um conjunto de orientações (ângulos para rotação dos itens), determinar as transformações (translação e orientação) aplicadas a cada item que os posicione completamente dentro do contêiner de modo a minimizar a função sobreposição. A [Figura 37](#) mostra um exemplo de problema de minimização de sobreposição e uma das soluções possíveis.

São fornecidos um contêiner retangular $\mathcal{C}$ de dimensões fixas, uma lista de polígonos $\mathcal{P} = \{P_1, \dots, P_n\}$, que representam os itens, e uma lista de ângulos $\mathcal{O}$ que corresponde às orientações admissíveis para os itens. Um polígono $P_i \in \mathcal{P}$ rotacionado de o é representado por $P_i(o)$. O objetivo é encontrar o vetor de translações $\mathbf{x}$ e o vetor de orientações $\mathbf{o}$, que são a solução do problema que segue:

$$\begin{aligned}
& \text{minimize} && \sum_{i=1}^n \sum_{j=1, j \neq i}^n \mathcal{F}(P_i(o_i), P_j(o_j)) \\
& \text{sujeito a} && (P_i(o_i) \oplus \mathbf{x}_i) \subseteq \mathcal{C} \quad 1 \leq i \leq n \\
& && L \in \mathbb{R}_+ \\
& && o_i \in \mathcal{O} \quad 1 \leq i \leq n \\
& && \mathbf{x}_i \in \mathbb{R}^2 \quad 1 \leq i \leq n
\end{aligned} \tag{4.1}$$

onde $\mathcal{F}(P_i(o_i), P_j(o_j))$ é a função sobreposição para os itens $P_i(o_i)$ e $P_j(o_j)$. Deve-se garantir, ao definir $\mathcal{F}$, que a seguinte condição é verdadeira:

$$\mathcal{F}(P_i(o_i), P_j(o_j)) = \mathcal{F}(P_j(o_j), P_i(o_i)). \tag{4.2}$$

Quando $\mathcal{F}$ é zero, uma solução válida foi encontrada. Neste problema adaptado, é possível gerar um espaço de solução mais conexo e, por este motivo, as técnicas de otimização tendem a obter melhores resultados.

4.3 Detecção e avaliação de sobreposição

A detecção de sobreposição é realizada utilizando o conceito de polígono de obstrução, descrito na [subseção 3.3.1](#), e o polígono de posicionamento interno (ver [subseção 3.3.2](#)) é utilizado para verificar se os itens estão completamente inseridos no contêiner. A utilização de ambas as ferramentas possibilita a verificação da validade de um leiaute. No entanto, para o problema de minimização de sobreposição, é necessário também definir a função sobreposição, cujo valor deve ser minimizado. Ademais, para esta proposta, é utilizado um método rasterizado para determinar o valor da função sobreposição para qualquer configuração dos itens. Estas ferramentas geométricas são descritas nesta seção.

4.3.1 Função sobreposição

A função sobreposição possui um grande impacto na solução de problemas de minimização de sobreposição. A eficiência do método de busca local é diretamente influenciada pela função sobreposição que, se definida de forma inconsistente, pode aumentar a

aleatoriedade da busca. Assim, considerações especiais devem ser feitas na definição da função sobreposição.

A função sobreposição é definida para um par de itens e é utilizada para relaxar a restrição de sobreposição. Desta forma, a condição para a definição da função é: o valor da função sobreposição deve ser zero quando os itens estão separados ou encostados, caso contrário, seu valor deve ser positivo. Utilizando apenas esta condição, já é garantida a convergência para uma solução válida quando o objetivo é minimizado, ou seja, quando o valor da soma de todas as funções é zero. Este é a única condição necessária para a aplicação do método baseado em minimização de sobreposição. No entanto, aspectos práticos geram a necessidade de estabelecer algumas diretrizes gerais para a definição de uma função sobreposição de modo que o emprego de técnicas de otimização seja favorecido. O seguinte comportamento é desejado na busca da solução do problema:

- quando um “esforço” muito grande é necessário para separar os itens, o valor da função deve ser alto;
- quando é necessário um “esforço” pequeno, o valor da função deve ser baixo;
- quando as funções são somadas, os pares mais difíceis de serem separados devem exercer maior influência.

Em linhas gerais, a função objetivo do problema deve representar a proximidade da configuração atual em relação à solução válida.

A medida mais intuitiva para a função sobreposição é a área da intersecção entre os itens, que foi empregada em (EGEBLAD; NIELSEN; ODGAARD, 2007). Em um problema de empacotamento, um par de itens com uma pequena área de sobreposição pode ser separado geralmente sem causar grandes modificações na solução, enquanto itens que se interseccionam em uma área grande podem demandar mais “esforço” para encontrar uma nova configuração sem gerar mais sobreposição no leiaute. No entanto, é possível encontrar casos em que esta constatação é falsa, como no posicionamento dos itens mostrados na Figura 38. No exemplo, os pares podem ser separados ao aplicar uma translação relativamente pequena a um dos itens. Todavia, devido à geometria dos itens, a área de intersecção é alta. Ademais, considerando o mesmo par de itens, é possível obter configurações igualmente próximas de serem separadas que apresentam valores bem distintos de área de sobreposição. Este fenômeno pode ser observado a partir da comparação dos pares de leiautes iguais da Figura 38.

4.3.1.1 Profundidade de penetração

A profundidade de penetração é outro valor empregado na literatura (IMAMICHI; YAGIURA; NAGAMOCHI, 2009; ELKERAN, 2013) para representar a função sobreposi-

Figura 38 – Área de sobreposição entre itens

Fonte: o autor

ção. Dado um par de itens P_i e P_j , a profundidade de penetração é a norma do menor vetor de translação que, aplicada a algum dos itens, obtém uma configuração sem sobreposição. A função penetração $\delta(P_i, P_j)$ pode ser descrita como:

$$\delta(P_i, P_j) = \min \{ \|v\| \mid i(P_j \oplus v) \cap i(P_i) = \emptyset \}. \quad (4.3)$$

É possível notar que, em concordância com a definição, $\delta(P_i, P_j) = \delta(P_j, P_i)$. Um exemplo de determinação da profundidade de penetração pode ser verificado na [Figura 39](#).

Figura 39 – Exemplo de profundidade de penetração

Fonte: o autor

A profundidade de penetração possui melhor correspondência com as necessidades de uma solução para o problema de empacotamento. É possível observar no exemplo da [Figura 38](#) que, apesar de possuírem um alto valor de área de sobreposição, os pares apresentam valores de profundidade de penetração relativamente baixos. Desta forma, observando apenas os valores da profundidade de penetração, é possível inferir que as configurações estão próximas da solução válida, o que pode ser visualmente confirmado. Por este motivo, a profundidade de penetração é adotada neste trabalho.

Uma vez que existe uma proximidade conceitual entre a profundidade de penetração e as ferramentas de detecção de sobreposição, é natural estender estes conceitos para a determinação da função sobreposição. O valor pode ser obtido utilizando a trigonometria direta, contudo, como já foi observado, outros métodos mais eficientes existem. O método rasterizado, que converte a geometria do item em uma matriz, é muito eficiente para o cálculo da área de sobreposição. No caso da profundidade de penetração, um procedimento mais complexo é exigido. Ademais, devido à baixa precisão, este método geralmente não é empregado. Assim como no caso da escolha de ferramenta geométrica para detecção de sobreposição, o polígono de obstrução é, na maior parte das vezes, o conceito mais adequado para se determinar o valor da profundidade de penetração.

Quando o polígono de obstrução é empregado, o problema de determinar se dois itens estão separados, encostados ou em sobreposição é transformado na verificação da posição de um ponto em relação a um polígono. Se o ponto estiver no contorno ou na região externa ao polígono, os itens estão separados. Utilizando esta propriedade, é possível definir uma metodologia para obtenção da profundidade de penetração baseada no conceito de polígono de obstrução. A profundidade de penetração de um par de itens P_i e P_j é a menor distância entre o ponto de referência r_i , correspondente ao item P_i , e algum ponto do contorno do polígono de obstrução $\Upsilon(P_i, P_j)$, e pode ser descrita como:

$$\delta(P_i, P_j) = \min \{ \|v\| \mid r_i + v \notin i(\Upsilon(P_i, P_j)) \}. \quad (4.4)$$

Devido a propriedade comutativa, o mesmo valor de profundidade de penetração também pode ser obtida a partir do polígono de obstrução induzido pelo item P_i . Assim, a mesma profundidade de penetração pode ser descrita como:

$$\delta(P_j, P_i) = \min \{ \|v\| \mid r_j + v \notin i(\Upsilon(P_j, P_i)) \}. \quad (4.5)$$

Desta forma, é possível utilizar o polígono de obstrução para determinar a profundidade de penetração. Neste caso, o problema é convertido no cálculo da menor distância do ponto até o contorno de um polígono.

4.3.1.2 Diagrama de Voronoi

Durante a execução da busca local, os cálculos geométricos de detecção de sobreposição envolvem a verificação da posição de um ponto em relação a um polígono. Este cálculo é relativamente simples e, deste modo, é aceitável o seu emprego em uma operação que é repetida diversas vezes. A fim de determinar a profundidade de penetração, deve-se determinar a menor distância entre um ponto e o contorno de um polígono, que é uma tarefa relativamente mais complexa. O método mais simples envolve, para cada aresta do polígono de obstrução, o cálculo da distância entre ponto e segmento de reta. Além

disso, quando o polígono não é convexo, mais casos devem ser considerados, uma vez que a distância do ponto de referência até um dos vértices não convexos pode ser menor do que a distância em relação à qualquer uma das arestas.

Uma estratégia adotada para reduzir o volume de processamento na obtenção da profundidade de penetração consiste em utilizar o diagrama de Voronoi. O diagrama de Voronoi aplicado a polígonos fechados, que também pode ser denominado de eixo médio, representa o conjunto de pontos que contém múltiplas posições mais próximas no contorno do polígono.

A Figura 40 apresenta um exemplo de diagrama de Voronoi aplicado a um polígono fechado. É possível observar que, no caso de polígono convexo, o diagrama de Voronoi realiza a divisão em regiões formadas por uma aresta do polígono e pelo menos duas linhas do diagrama. A menor distância em relação ao contorno do polígono de qualquer ponto no interior de uma região pode ser calculada a partir da determinação da distância deste ponto em relação à aresta do polígono da região. Este procedimento é ilustrado na Figura 40. Os pontos p_1 , p_2 e p_3 estão contidos na região R_1 , de modo que a menor distância para algum ponto do polígono é obtida a partir do cálculo da distância do ponto até a aresta a_1 , que compõe a região R_1 . Quando um ponto se encontra sobre alguma linha do diagrama de Voronoi, como no caso do ponto p_4 no exemplo, a menor distância pode ser definida com o auxílio de qualquer uma das regiões delimitadas pela linha (R_3 e R_4). No exemplo, a distância procurada para o ponto p_4 é obtida a partir das arestas a_3 ou a_4 .

Figura 40 – Diagrama de Voronoi de um polígono convexo.

Fonte: o autor

Ao aplicar o diagrama de Voronoi ao polígono de obstrução, o procedimento descrito para determinar a menor distância entre um ponto e o contorno do polígono resulta no valor da profundidade de penetração. Por conseguinte, o cálculo da função sobreposição é convertido, primeiramente, na determinação da posição do ponto de referência em relação às regiões geradas pelo diagrama de Voronoi, seguido do cálculo da distância entre ponto e reta. Esta metodologia pode ser aplicada nos casos em que o polígono de obstrução é convexo. Caso contrário, adaptações devem ser realizadas, o que pode aumentar a

Figura 41 – Montanha Voronoi de um polígono convexo

Fonte: o autor

complexidade do método. Em geral, o diagrama de Voronoi para polígonos não convexos contém, além de linhas, curvas descritas por polinômios de grau maior do que um e, por este motivo, sua utilização pode ser ineficiente. Assim, outra possibilidade é utilizar uma aproximação para gerar o diagrama de Voronoi, o que, no entanto, pode causar problemas de precisão.

4.3.1.3 Montanha Voronoi

Mesmo utilizando o diagrama de Voronoi, ainda é necessário, para cada cálculo da profundidade de penetração, realizar a determinação da distância ponto-reta, além da definição da posição relativa do ponto de referência. A fim de reduzir a necessidade de cálculos repetidos, o conceito de montanha Voronoi pode ser utilizado. Este conceito foi introduzido visando a solução do problema de busca pelo caminho ótimo de corte em usinagem 2.5D (VEERAMANI; GAU, 1997). É proposta neste trabalho a utilização deste conceito em problemas de minimização de sobreposição. O objetivo é diminuir a carga computacional do algoritmo de determinação da profundidade de penetração, que deve ser utilizado reiteradamente durante um processo de busca local.

A montanha Voronoi é uma extensão do conceito do diagrama de Voronoi e consiste em uma superfície construída a partir das regiões definidas pelo diagrama. A altura de um ponto da superfície da montanha Voronoi corresponde à menor distância entre a projeção do ponto no plano que contém o polígono original e o contorno do mesmo, como pode ser observado na Figura 41. Quando a geometria do polígono é convexa, a montanha Voronoi é a composição de planos delimitados pelas regiões do diagrama de Voronoi. O ângulo deste plano em relação ao plano que contém o polígono é de 45° . Considere que o polígono original está contido no plano xy e a coordenada z representa a altura. Uma região do diagrama de Voronoi contém a aresta definida pelos pontos $P_1 = (P_{1x}, P_{1y})$ e

$P_2 = (P_{2x}, P_{2y})$. O plano correspondente a esta região é dado pela equação

$$z = \frac{1}{\Delta^2} [\Delta_y \cdot x + \Delta_x \cdot y - \Delta^2] \quad (4.6)$$

onde $\Delta^2 = \sqrt{\Delta_y^2 + \Delta_x^2}$, $\Delta_x = P_{2x} - P_{1x}$ e $\Delta_y = P_{2y} - P_{1y}$. Isto posto, como a [Equação 4.6](#) equivale à distância ponto-reta, a altura da montanha Voronoi representa a menor distância do ponto até o contorno do polígono.

No caso de polígonos não convexos, além dos planos, a superfície da montanha Voronoi é formada por cones invertidos, que são centrados nos vértices não convexos e cuja geratriz forma um ângulo de 45° com o seu eixo. Considere $P_c = (P_{cx}, P_{cy})$ o vértice associado a uma região definida pelo diagrama de Voronoi de um polígono não convexo. O cone gerado é dado pela equação

$$z = \sqrt{(x - P_{cx})^2 + (y - P_{cy})^2} \quad (4.7)$$

que é equivalente à equação de distância entre pontos e, por este motivo, representa a menor distância do ponto P_c até o contorno do polígono.

Como é possível observar a partir da [Equação 4.7](#), quando polígonos não convexos são considerados, termos de segundo grau devem ser considerados. Combinada com a complexidade da obtenção das regiões do diagrama de Voronoi, a aplicabilidade destes conceitos para problemas de empacotamento que envolvem polígonos de obstrução côncavos é bastante reduzida se alguma medida de simplificação não é adotada. Neste trabalho, o método rasterizado é adaptado a fim de reduzir a complexidade da aplicação da montanha Voronoi na determinação da profundidade de penetração.

4.3.2 Transformada distância

A proposta deste trabalho consiste em limitar o espaço de busca, que é originalmente infinito, através da utilização de uma grade para o posicionamento dos itens. Desta forma, considerando apenas dois itens, existe apenas um número finito de configurações de sobreposição. Estas posições podem ser mapeadas para uma região no espaço a partir da realização da rasterização do polígono de obstrução. Desta forma, a detecção de sobreposição pode ser realizada consultando apenas um elemento da matriz que representa o polígono de obstrução discretizado.

A [Figura 42](#) mostra um exemplo de detecção de sobreposição utilizando o polígono de obstrução rasterizado. Assim, como em outras soluções que admitem o posicionamento em uma grade, em geral não é possível obter todas as posições em que os itens se encostam. Nestas situações, existem duas alternativas: admitir sobreposição ou deixar uma lacuna

entre os itens. O erro produzido pode ser atenuado reduzindo as dimensões da grade de rasterização.

Figura 42 – Polígono de obstrução rasterizado

Fonte: o autor

Figura 43 – Montanha de penetração discreta

Fonte: o autor

Outra vantagem na adoção de métodos rasterizados é a possibilidade de utilização de técnicas de operações com matrizes. Na maioria das vezes, essas operações são mais simples que as equivalentes geométricas. Para a determinação da sobreposição utilizando o polígono de obstrução, no caso poligonal, é necessária a averiguação da posição de um ponto em relação a um polígono e, no caso rasterizado, apenas a verificação de um elemento da matriz é suficiente. Apesar da redução da complexidade das operações, o volume de dados, que é determinado pelo tamanho da grade de rasterização, é o fator mais influente no desempenho dos algoritmos.

A fim de obter o valor da profundidade de penetração, uma estratégia bastante eficiente requer a obtenção da montanha Voronoi de cada polígono de obstrução. Entretanto, este cálculo é complexo para polígonos de obstrução não convexos. Ademais, é difícil obter uma representação eficiente para o diagrama de Voronoi, que serve de base para a montanha Voronoi, uma vez que esta pode ser formada a partir de polinômios de grau maior que um. Por estes motivos, é introduzido neste trabalho o conceito de montanha de penetração discreta, que utiliza o polígono de obstrução rasterizado para obter a profundidade de penetração. O objetivo é obter o valor de profundidade de penetração para cada pixel interno ao polígono de obstrução rasterizado. Um exemplo deste mapeamento pode ser observado na [Figura 43](#).

A transformada distância é uma técnica utilizada para processamento de imagens, muitas vezes adotada para determinar o esqueleto de uma imagem binária, que é similar ao eixo médio para polígonos. A entrada de um algoritmo de transformada distância consiste em uma imagem binária, cujos pixels possuem duas classificações possíveis: pixel obstáculo ou pixel livre. A transformada distância atribui valor aos pixels livres de acordo com a sua distância em relação ao pixel obstáculo mais próximo. A definição da formulação da distância pode ser modificada de acordo com a necessidade. Quando a distância euclidiana é

utilizada e o algoritmo é aplicado a um polígono de obstrução discretizado, os elementos da matriz resultante possuem valor proporcional à profundidade de penetração. As distâncias são processadas em unidades de pixels na transformada distância e, deste modo, o valor real da profundidade de penetração é facilmente obtido a partir do tamanho da grade.

Figura 44 – Passos para a obtenção da montanha de penetração discreta

Fonte: o autor

A [Figura 44](#) exibe um exemplo do procedimento de obtenção da profundidade de penetração utilizando o polígono de obstrução discretizado, cujo resultado é a montanha de penetração discreta. No exemplo, as três etapas para a obtenção da montanha de penetração discreta são ilustradas: a obtenção do polígono de obstrução, a rasterização e a transformada distância. Outros dois métodos podem ser definidos para obter a montanha de penetração discreta. O primeiro consiste em aplicar a rasterização diretamente à montanha Voronoi. Como já discutido antes, o processamento geométrico é deveras complexo. A última opção é aplicar a rasterização na geometria original do polígono, obter o polígono de obstrução discreto diretamente e aplicar a transformada distância. Neste caso, o problema de precisão é mais crítico, uma vez que não é possível determinar exatamente as posições de sobreposição da grade no polígono de obstrução. Isto posto, o procedimento ilustrado na [Figura 44](#) é o escolhido para a geração da montanha de penetração discreta, que é a ferramenta geométrica adotada para a determinação da profundidade de penetração.

4.4 Minimização de sobreposição

A minimização de sobreposição consiste na solução do problema descrito pela [Equação 4.1](#), cuja solução ótima é um leiaute válido, no qual o valor da função objetivo é zero. Em geral são empregados algoritmos baseados em busca local e uma meta-heurística é adotada a fim de evitar que a mesma fique presa em um ótimo local. Uma vez encontrada uma solução que não apresenta sobreposição, o algoritmo é interrompido e a execução pode ser considerada bem sucedida. Se alguma outra condição de parada é atingida, não foi possível encontrar uma solução válida e, portanto, a busca deve ser considerada falha.

4.4.1 Mapa de sobreposição total

Abordagens para o problema da minimização da sobreposição geralmente utilizam algum método de busca local que altera a solução iterativamente. Na literatura, é possível verificar que, em propostas que movimentam um item por vez, geralmente três tipos de operadores são utilizados para modificar o leiaute. O primeiro consiste em trocar a posição de dois itens. Já o segundo e o terceiro, que são muitas vezes empregados em conjunto, consistem na alteração da posição e na definição de uma nova orientação para o item, respectivamente. No grupo de abordagens que utilizam a minimização de sobreposição, os últimos dois operadores são mais comumente adotados.

O operador que troca dois itens de posição é útil nos casos em que não existe informação suficiente para determinar uma boa posição para o item. Se o leiaute inicial possui um valor baixo de sobreposição, a nova configuração, resultante desta operação, não apresenta um valor muito maior de sobreposição. Esta situação ocorre especialmente no caso em que itens de dimensões próximas são escolhidos, como pode ser observado na [Figura 45](#). No entanto, é complexo definir quais itens devem ser escolhidos a fim de obter um leiaute com menor valor de sobreposição e, por este motivo, muitas vezes esta escolha é randômica.

Figura 45 – Troca de dois itens no leiaute

Fonte: o autor

Pelos motivos apresentados, é possível concluir que, quando aplicável, a utilização dos operadores que definem uma nova posição e orientação deve ser privilegiada. A estratégia para a definição destes parâmetros é importante para determinar a eficiência do processo de busca. Em propostas para a solução da minimização da sobreposição, esta abordagem é amplamente utilizada. Nestes casos, a função sobreposição é utilizada para realizar a movimentação do item. Como este valor considera apenas um par de itens, é definida uma nova função composta por todas as funções sobreposição que envolvem o

item. A função sobreposição total para um item P_i pode ser descrita como

$$F_i(x, o) = \sum_{j=1, j \neq i}^n \delta(P_i(o_i), P_j(o_j)). \quad (4.8)$$

A partir desta definição, é possível gerar o operador que realiza o movimento de um item de modo a minimizar o valor de sua sobreposição total. Se a sobreposição total de cada item for reduzida, a função objetivo será consequentemente menor. Uma vez que a movimentação é realizada de forma sequencial, não há garantia que, a cada iteração, a função sobreposição total dos outros itens do leiaute seja modificada de modo a aumentar a soma das sobreposições. Mesmo assim, esta estratégia foi implementada com sucesso na literatura, o que indica que, se uma estratégia adequada é adotada, o valor da função objetivo é geralmente reduzido.

O operador de inserção, na maior parte das vezes, limita o posicionamento dos itens de acordo com o polígono de posicionamento interno, garantindo que o item esteja totalmente contido no contêiner. Mesmo com esta limitação, que consiste em quatro restrições tipo caixa, a definição de F_i para todas as posições é bastante custosa. Ademais, a busca nessa região também pode ser bastante ineficiente, especialmente no caso da presença de polígonos de obstrução côncavos. Por conseguinte, as soluções na literatura utilizam estratégias que reduzem o espaço de busca.

Neste trabalho, é utilizada a discretização do posicionamento dos itens, o que possibilita o cálculo da função sobreposição total para todas as configurações possíveis. O resultado deste procedimento é denominado mapa de sobreposição total, que é um conceito proposto neste trabalho. Um exemplo de mapa de sobreposição total pode ser observado na [Figura 46](#). O mapa de sobreposição total pode ser obtido a partir da soma das montanhas de penetração discretas, deslocadas de acordo com a posição dos itens no leiaute. A utilização do mapa de sobreposição possibilita a determinação da nova posição de um item no leiaute.

4.4.2 Descrição da busca local

Em uma iteração da busca local, os itens são movimentados individualmente utilizando o conceito de mapa de sobreposição total para definir a sua melhor posição no leiaute. Uma vez que o leiaute é modificado, é conveniente combinar o movimento de translação com a aplicação do operador de rotação, que escolhe uma nova orientação para o item. Desta forma, o impacto da aplicação de dois operadores é amenizado. Do mesmo modo como é conduzida a operação de translação, é apropriado utilizar o valor da sobreposição para definir a nova rotação para o item. Assim, o par translação/rotação escolhido é aquele que minimiza o valor da função sobreposição total.

Figura 46 – Exemplo de mapa de sobreposição total

Fonte: o autor

A sequência na qual os itens são movidos exerce um impacto relativamente baixo no leiaute a ser obtido. No entanto, é mais provável que a busca local fique presa em um ótimo local caso uma sequência estática seja adotada, pois a busca pode resultar em um movimento cíclico dos itens no leiaute. Por este motivo, a sequência de movimentação é gerada aleatoriamente a cada iteração.

Algoritmo 2 BUSCALOCAL

```

 $n \leftarrow$  número de itens
 $Q \leftarrow$  sequência aleatória dos itens
Para cada item  $P_k \in Q$  faça
  MínSobreposição =  $+\infty$ 
  Para cada orientação permitida  $o_l$  faça
    <Constrói mapa de sobreposição do item  $P_k(o_l)$ >
     $val \leftarrow$  <menor valor do mapa>
     $pos \leftarrow$  <posição do menor valor do mapa>
    Se  $val < \text{MínSobreposição}$  então
       $x_k = pos$ 
      MínSobreposição  $\leftarrow val$ 
    Fim Se
  Fim Para
Fim Para
  
```

O Algoritmo 2 descreve o algoritmo da busca local adotada. A Figura 47 apresenta um exemplo de execução de iterações da busca local até a obtenção de uma solução válida.

Figura 47 – Sequência de execuções da busca local

Fonte: [Elkeran \(2013\)](#)

4.4.3 Busca local guiada

A busca local pode ser aplicada diretamente para eliminar a sobreposição do leiaute. Ao final de cada iteração, a função objetivo é analisada. Se o seu valor for menor que o correspondente à última solução aceita, o novo resultado é aceito. Caso contrário, a solução é descartada. Seguindo este procedimento, são aceitas apenas soluções melhores que as já obtidas. Esta estratégia possui a tendência de convergir rapidamente para um ótimo local e não é capaz de gerar outra solução que possibilite o prosseguimento da busca.

Meta-heurísticas possuem a capacidade de escapar de ótimos locais, com o intuito de resumir a busca pelo ótimo global do problema. Isto é obtido a partir da aceitação de algumas soluções que não melhoram o resultado, de acordo com algum critério. Desta forma, uma vez que um ótimo local é encontrado, a heurística de busca deve permitir que uma solução menos eficiente seja aceita, com o propósito de prosseguir a busca e encontrar uma melhor solução.

Meta-heurísticas podem definir regras a serem seguidas na definição da busca local de modo a evitar que sua execução fique estacionada em uma solução ótima apenas localmente. Várias meta-heurísticas foram empregadas para resolver o problema de empacotamento irregular de fita. Meta-heurísticas mais genéricas como a busca tabu, recozimento simulado e algoritmos genéticos são as mais aplicadas ao empacotamento irregular de fita, uma vez que estas são muito eficientes para situações em que não é possível extrair muita informação sobre o problema. Porém, quando uma função sobreposição é adotada, o comportamento do problema é mais conhecido. Assim, nestes casos, é conveniente utilizar uma meta-heurística que melhor adapta as características do subproblema de minimização de sobreposição.

A busca local guiada é uma meta-heurística muitas vezes empregada para solucionar o problema da minimização da sobreposição. Neste trabalho, é adotada a proposta de ([UMETANI et al., 2009](#)). Para a aplicação da busca local guiada, a função objetivo deve ser composta por parcelas, que representam informações locais do estado da solução.

Pesos são atribuídos a cada parcela, gerando uma nova função objetivo. Os pesos são atualizados a cada iteração, possibilitando o controle da exploração do espaço de soluções, de modo a evitar que a busca fique presa em um ótimo local.

No caso estudado, o valor da função objetivo é obtido a partir das profundidades de penetração individuais, que estão relacionadas ao posicionamento de dois itens. Desta forma, pesos associados a cada par de itens são incluídos na função sobreposição total modificada F'_i , que pode ser descrita como

$$F'_i(x, o) = \sum_{j=1, j \neq i}^n w_{ij} \cdot \delta(P_i(o_i), P_j(o_j)). \quad (4.9)$$

A cada iteração, os pesos são modificados de acordo com o valor atual da profundidade de penetração do par de itens correspondente. Este montante é normalizado e acrescido ao valor do peso. Sendo assim, os pesos são atualizados de acordo com a seguinte regra

$$w_{ij} = w_{ij} + \frac{\delta(P_i, P_j)}{\max(\delta(P_k, P_l))} \quad (4.10)$$

onde $1 \leq k, l \leq n$. É possível observar que, quando muitas soluções iguais ou muito próximas são geradas em sequência, ou seja, um ótimo local é encontrado, o mesmo conjunto de pesos é atualizado. Isto impacta a movimentação dos itens atualmente em estado de sobreposição de modo que, à medida que novos leiautes são gerados, aumenta a tendência de escapar do ótimo local. A [Figura 48](#) mostra um exemplo desta situação. Na ilustração, o leiaute inicial consiste em um ótimo local encontrado pelo algoritmo e o mapa de sobreposição do item central é exibido. Conforme os pesos são modificados, o mapa do item é alterado de modo a permitir a movimentação do item após três iterações. A partir desta configuração, novas soluções serão geradas, o que resume a busca pela melhor solução.

A vantagem da utilização da busca local guiada em relação a meta-heurísticas mais genéricas é a possibilidade de escapar de ótimos locais através da aplicação de modificações locais no leiaute, não alterando regiões em que a configuração dos itens é eficiente. Utilizando a busca local guiada, é possível definir o [Algoritmo 3](#), utilizado para resolver o problema de minimização de sobreposição.

4.5 Compactação do leiaute

Assim como em outras soluções que adotam o contêiner de dimensões invariáveis para resolver o empacotamento de fita, a compactação do leiaute é uma camada externa do algoritmo e é responsável por controlar o comprimento do contêiner. O [Algoritmo 3](#) é

Figura 48 – Fuga do ótimo local utilizando a busca local guiada

Leiaute:

Mapa de sobreposição total do item central:

Fonte: o autor

Algoritmo 3 MINIMIZASOBREPOSIÇÃO

```

it ← 0
aptidao = +∞
Enquanto it < Nmr faça
  <Executa BUSCALOCAL (Algoritmo 2) utilizando pesos>
   $val = \sum_{i=1}^n \sum_{j=1, j \neq i}^n \mathcal{F}(P_i(o_i), P_j(o_j))$ 
  Se val = 0 então
    retorna (x,o)
  Senão [val < aptidao]
    aptidao = val
    it = 0
  Fim Se
  it ← it + 1
  <Atualiza pesos>
Fim Enquanto

```

---

utilizado como o nível interno, que é encarregado de obter um leiaute sem sobreposição dado um contêiner de comprimento fixo.

O Algoritmo 4 detalha o processo de compactação do leiaute utilizado nesta proposta. Assim como no método descrito na subseção 3.5.2, dois parâmetros são definidos para determinar a taxa de redução (p_{red}) e expansão (p_{exp}) do comprimento do contêiner.

Algoritmo 4 RESOLVEEMPACOTAMENTO

```

 $L_c \leftarrow \langle \text{Comprimento inicial do contêiner} \rangle$ 
Enquanto Não atingiu o tempo limite faça
   $\langle \text{Executa o algoritmo de minimização de sobreposição} \rangle$ 
  Se Leiaute válido foi encontrado então
     $L_c \leftarrow (1 - p_{red}) \cdot L_c$ 
  Senão
     $L_c \leftarrow (1 + p_{exp}) \cdot L_c$ 
  Fim Se
   $\langle \text{Comprimento do contêiner} \rangle \leftarrow L_c$ 
Fim Enquanto

```

As estratégias se distinguem no tratamento dos itens que devem ser realocados após uma redução do contêiner. No [Algoritmo 4](#), os itens são reposicionados no contêiner utilizando o mesmo operador de inserção adotado na busca local. Esta regra visa reduzir o número de operações que modificam o leiaute. A [Figura 49](#) exibe um exemplo deste processo.

Figura 49 – Compactação do leiaute com a admissão de sobreposição

Fonte: [Imamichi, Yagiura e Nagamochi \(2009\)](#)

4.5.1 Solução inicial

O comprimento inicial do contêiner deve ser definido a fim de começar o empacotamento dos itens. Não é simples obter uma boa estimativa deste valor analisando apenas dados gerais do problema, como o número de itens ou a soma de suas áreas, pois a geometria irregular dos itens exerce um grande impacto na geração de leiautes sem sobreposição. Assim, é prudente que a escolha do comprimento inicial do contêiner seja

definida a partir da criação de um leiaute válido. A definição da solução inicial deve ser rápida, a fim de disponibilizar mais tempo para a fase de empacotamento, e é desejável que sua densidade seja a mais alta possível, iniciando o processo de compactação em uma etapa mais avançada.

Na literatura é possível encontrar três estratégias principais para obter um leiaute válido: através do relaxamento da restrição de sobreposição, do emprego de métodos matemáticos ou da utilização de uma heurística de posicionamento construtiva. A primeira opção requer uma solução inicial válida e, portanto, pode ser desconsiderada. Modelos matemáticos de separação e compactação são bastante eficientes para melhorar a compactação de soluções válidas. No entanto, não são geralmente adequados para gerar uma solução inicial, pois requerem maior tempo para processar uma solução aleatória. Deste modo, a opção mais adequada a ser empregada neste caso é a utilização de alguma heurística construtiva. Devido a sua capacidade de gerar leiautes de boa compactação em um tempo bastante reduzido, a heurística inferior esquerdo foi escolhida para ser empregada nesta proposta.

Devido à limitação do posicionamento através da grade, a geração do leiaute inicial também deve considerar coordenadas discretas para os itens. Por este motivo, foi desenvolvido um algoritmo de construção de leiaute que compartilha as ferramentas geométricas rasterizadas adotadas para o empacotamento. A heurística de posicionamento adotada posiciona o primeiro item na coordenada inferior esquerda do polígono de posicionamento interno. Para os demais itens, é feita uma sequência de varreduras verticais, de baixo para cima, que inicia na coordenada horizontal correspondente à parede esquerda do contêiner e prossegue na direção positiva do eixo horizontal. Caso um valor nulo de sobreposição total é encontrado, o item é posicionado. O procedimento é repetido até que todos os itens da sequência estejam inseridos no leiaute. É adotada uma sequência aleatória de posicionamento, determinada no início de cada execução.

Uma vez gerada a solução inicial válida, o contêiner é dimensionado de modo que o seu comprimento seja o mínimo capaz de acomodar todos os itens do leiaute.

4.6 Extensão do método

A adoção de métodos para lidar com a representação rasterizada visa converter o problema de processar cálculos geométricos complexos com polígonos em operações simples com matrizes, que podem possuir muitos elementos. O grande volume de dados, que é regulado pela precisão desejada para a solução, pode prejudicar a aplicabilidade do método. Desta forma, são investigadas estratégias para melhorar o desempenho do algoritmo, a fim de possibilitar a solução do problema com maior precisão.

Na solução utilizando métodos rasterizados, é proposto o conceito de mapa de

sobreposição total. Na busca local adotada, cada iteração consiste na movimentação dos itens utilizando este mapa. Esta operação é repetida para todos os itens a cada iteração. Ademais, quando múltiplas orientações são admitidas, deve ser criado um mapa de sobreposição total para cada rotação. A geração e a avaliação do mapa de sobreposição total são, na maior parte dos casos, as tarefas mais custosas computacionalmente. Isto posto, são propostos dois métodos para reduzir o tempo computacional requerido para obter e processar os mapas: a utilização de múltiplas resoluções e a paralelização massiva em placas gráficas. Além disso, é também introduzido um método paralelo para a obtenção da montanha de penetração discreta.

4.6.1 Abordagem com múltiplas resoluções

A precisão geométrica da solução é determinada a partir do tamanho da grade utilizada para o posicionamento. Caso pequenas sobreposições sejam permitidas, a precisão está relacionada à maior área de sobreposição admitida. Caso contrário, a precisão diz respeito ao espaço desperdiçado entre itens devido à rasterização.

As dimensões da grade de posicionamento são utilizadas para definir a rasterização da montanha Voronoi do polígono de obstrução. Deste modo, a verificação de sobreposição é realizada em coordenadas da grade.

Considere que a resolução do mapa de sobreposição é dada pelo número de elementos da grade por unidade das coordenadas do polígono original. Assim, a resolução do mapa de sobreposição total, que é composto pela soma das montanhas de penetração discretas, é proporcional ao quadrado da dimensão da grade, que por sua vez é definida pela precisão adotada.

A complexidade do algoritmo de geração e de busca pela posição por movimentação do item é $O(n^2)$, onde n é o tamanho da grade quadrada, que é inversamente proporcional à precisão. É possível concluir que o aumento da precisão tem grande impacto no desempenho do algoritmo. Muitas vezes, resoluções baixas devem ser adotadas a fim de possibilitar uma compactação satisfatória em um tempo computacional razoável.

Com o intuito de possibilitar a solução do problema com maior precisão, é proposta a utilização de duas resoluções para o mapa de sobreposição total. A proposta consiste em utilizar resoluções diferentes para o posicionamento de itens e para o mapa de sobreposição total. Devido ao desalinhamento das grades e possibilidade de refinamento do posicionamento, é necessária uma etapa adicional para determinar a nova posição para um item, que consiste na geração de um mapa de sobreposição total de dimensões reduzidas (vide [Figura 50](#)). Este procedimento possui custo computacional relativamente baixo. Desta forma, o desempenho é condicionado à discretização adotada para o mapa de sobreposição total, que deve possuir a menor das duas resoluções. A precisão geométrica é dada pela

Figura 50 – Obtenção da posição de menor sobreposição de um item utilizando a busca com duas resoluções

Fonte: o autor

grade de posicionamento, cuja resolução deve ser maior.

4.6.2 Paralelização massiva

Operações geométricas são muitas vezes ineficientes devido à interdependência dos dados, que é ocasionada pelo restrito relacionamento entre os elementos que representam a geometria do item, como vértices e arestas. Quando a representação rasterizada é adotada, operações de matrizes são utilizadas e geralmente estas operações tratam cada elemento individualmente. Isto possibilita o emprego de técnicas de paralelização a fim de diminuir o tempo computacional das operações.

4.6.2.1 Determinação da montanha de penetração discreta

A vantagem da utilização do polígono de obstrução para detectar sobreposição é a possibilidade de realizar o seu cálculo em uma etapa de pré-processamento. Uma vez que a montanha de penetração discreta é uma extensão do conceito de polígono de obstrução, esta propriedade também é válida. Para um dado problema, deve ser determinada uma

montanha de penetração para cada par de itens distintos e, caso múltiplas orientações sejam admitidas, cada rotação deve ser considerada como um item distinto. Deste modo, o número de montanhas de sobreposição a ser calculado pode ser relativamente alto, elevando os requisitos de memória e prejudicando o desempenho do algoritmo. É importante notar que o tempo de execução da etapa de pré-processamento varia de acordo com o número de itens distintos e orientações permitidas, não importando o número total de itens.

A montanha de penetração discreta é a ferramenta utilizada para determinar a função sobreposição entre itens. Esta é determinada utilizando a transformada distância, que é uma técnica de processamento de imagem. Este algoritmo recebe como entrada uma imagem obtida a partir da rasterização do polígono de obstrução. Esta imagem deve ser composta por dois tipos de pixels: obstáculos e livres. A transformada distância realiza o mapeamento da distância euclidiana entre um pixel livre e a região formada pelos pixels obstáculos.

4.6.2.1.1 Algoritmo 4SED (*Sequential Euclidean Distance*)

Danielsson (1980) propôs dois algoritmos para realizar o mapeamento da distância euclidiana: 4SED e 8SED. Neste trabalho foi adotado o algoritmo 4SED, que é mais rápido. O 4SED apresenta erros de aproximação maiores, entretanto, para o problema de minimização de sobreposição, este aspecto do algoritmo não possui um impacto tão significativo.

O Algoritmo 5 determina a transformada da distância a partir de uma imagem de dimensões (M, N) . Um mapa auxiliar $\bar{L}$, cujas células contêm um par de valores, é criado com o mesmo tamanho da imagem de entrada. O mapa é inicializado com valores $(0,0)$ se o pixel pertencer ao fundo (obstáculo) e com $(+\infty, +\infty)$ se pertencer ao objeto (livre). Os pixels obstáculos são propagados utilizando uma estrutura chamada modelos de vetor (*vector templates*). Cada vez que um pixel é processado, o melhor candidato dentro do modelo é escolhido. São feitas duas varreduras na imagem, propagando as mudanças a partir dos pixels obstáculos. Ao fim da segunda varredura cada célula do objeto contém as distâncias horizontal e vertical para o fundo. A transformada da distância é concluída computando a raiz quadrada da soma dos quadrados de cada elemento.

Os modelos de vetor indicam a direção da varredura e quais vizinhos serão analisados durante a varredura de um pixel. Os modelos de vetor utilizados no algoritmo 4SED estão representados na Figura 51. A Figura 52, onde $Z = +\infty$, mostra um exemplo de varredura em uma direção utilizando um modelo de vetor.

Algoritmo 5 4SED

Para todo pixel (i, j) da imagem **faça**
 $\bar{L}(i, j) \leftarrow (0, 0)$, se (i, j) pertence ao fundo
 $\bar{L}(i, j) \leftarrow (+\infty, +\infty)$, caso contrário
Fim Para
Para $j = 1$ até $N - 1$ **faça**
  Para $i = 0$ até $M - 1$ **faça**
     $\bar{L}(i, j), \bar{L}(i, j) \leftarrow \min(\bar{L}(i, j), \bar{L}(i, j - 1) + (0, 1))$
  Fim Para
  Para $i = 1$ até $M - 1$ **faça**
     $\bar{L}(i, j), \bar{L}(i, j) \leftarrow \min(\bar{L}(i, j), \bar{L}(i - 1, j) + (1, 0))$
  Fim Para
  Para $i = M - 2$ até 0 **faça**
     $\bar{L}(i, j), \bar{L}(i, j) \leftarrow \min(\bar{L}(i, j), \bar{L}(i + 1, j) + (1, 0))$
  Fim Para
Fim Para
Para $j = N - 2$ até 0 **faça**
  Para $i = 0$ até $M - 1$ **faça**
     $\bar{L}(i, j), \bar{L}(i, j) \leftarrow \min(\bar{L}(i, j), \bar{L}(i, j + 1) + (0, 1))$
  Fim Para
  Para $i = 1$ até $M - 1$ **faça**
     $\bar{L}(i, j), \bar{L}(i, j) \leftarrow \min(\bar{L}(i, j), \bar{L}(i - 1, j) + (1, 0))$
  Fim Para
  Para $i = M - 2$ até 0 **faça**
     $\bar{L}(i, j), \bar{L}(i, j) \leftarrow \min(\bar{L}(i, j), \bar{L}(i + 1, j) + (1, 0))$
  Fim Para
Fim Para

Figura 51 – Modelos de vetor do algoritmo 4SED

Fonte: o autor

4.6.2.1.2 Paralelização

O algoritmo 4SED (DANIELSSON, 1980) não pode ser paralelizado trivialmente, uma vez que a determinação do valor de uma célula do mapa influencia os cálculos posteriores da varredura. Também não é possível realizar várias varreduras em paralelo, dado que pode haver interferência entre elas.

A fim de possibilitar a paralelização do mapeamento da distância euclidiana, Schneider, Kraus e Westermann (2009) propuseram um novo conjunto de modelos de vetor.

Figura 52 – Exemplo de uma varredura do algoritmo 4SED

Fonte: o autor

O objetivo foi obter um conjunto que possibilitasse a realização de todas as varreduras em uma mesma direção em paralelo. Em compensação são necessárias mais varreduras para obter o resultado final. Para cada etapa, todas as varreduras em uma mesma direção devem ser finalizadas antes do prosseguimento do algoritmo. A Figura 53 exibe o conjunto de modelos de vetor adotado. O Algoritmo 6 mostra como os modelos de vetores devem ser utilizados para realizar o mapeamento das distâncias.

Figura 53 – Modelos de vetor do algoritmo paralelo

Fonte: o autor

Algoritmo 6 4SED PARALELIZAVEL

Para todo pixel (i, j) da imagem **faça**

$\bar{L}(i, j) \leftarrow (0, 0)$, se (i, j) pertence ao fundo

$\bar{L}(i, j) \leftarrow (+\infty, +\infty)$, caso contrário

Fim Para

Para cada modelo de vetor **faça**

Para todo pixel (i, j) da imagem **faça**

$(i, j) \leftarrow -(k, l)$, onde (k, l) é o pixel do modelo mais próximo de (i, j)

Fim Para

Fim Para

4.6.2.1.3 Implementação em CUDA

A paralelização do algoritmo de [Schneider, Kraus e Westermann \(2009\)](#) pode ser realizada utilizando processadores multinúcleos ou unidades de processamento gráfico (GPUs). Maior paralelização é obtida se utilizada a GPU, uma vez que possui um maior número de unidades de processamento. A proposta em ([SCHNEIDER; KRAUS; WESTERMANN, 2009](#)) utiliza GPU, porém foi implementada utilizando *shaders*. Neste trabalho foi escolhida a elaboração do código em *Compute Unified Device Architecture* (CUDA), suportada pelas unidades de processamento gráfico da NVIDIA, devido à maior facilidade de implementação e manutenção.

Na arquitetura CUDA, as funções na GPU (*kernels*) especificam o código a ser executado paralelamente por um elevado número de *threads* (processos). Ao ser lançada uma função *kernel*, automaticamente é gerada uma grade de execução (*grid*), organizada em sub-blocos de *threads*. *Threads* pertencentes a um mesmo bloco podem cooperar mais facilmente entre si, através de memórias compartilhadas e de operações de sincronização específicas.

4.6.2.2 Determinação do mapa de sobreposição total

O mapa de sobreposição total é gerado a partir da soma das montanhas de penetração deslocadas de acordo com a posição dos itens fixos. Como a representação de pixels é utilizada para descrever a geometria da montanha de penetração, é possível utilizar matrizes para armazenar as informações da função sobreposição. Cada elemento da matriz contém o valor da profundidade de penetração para a configuração relativa dos itens correspondentes. Operações de matrizes podem, então, ser utilizadas para construir o mapa de sobreposição total.

Na soma de matrizes, que deve ser adotada para determinar o mapa, cada elemento do resultado é obtido a partir da soma dos elementos de mesmo índice das parcelas. Deste modo, a paralelização pode ser realizada de forma direta, atribuindo cada *thread* a um elemento da matriz resultante. O número de elementos de uma matriz que representa o mapa de sobreposição total pode ser demasiadamente elevado, principalmente no caso em que alta precisão é adotada. Consequentemente, o desempenho do algoritmo paralelo depende do número de *threads* disponíveis, que favorece a utilização de paralelização massiva em GPUs. A busca pelo menor valor do mapa de sobreposição total, que é realizada para determinar translações para os itens, também pode ser processada paralelamente na GPU a partir de técnicas de paralelização de operações de redução, embora o ganho de processamento seja menor do que no caso da geração dos mapas de sobreposição total.

4.7 Resultados

Os resultados desta seção estão divididos em três partes. As primeiras duas partes relatam os resultados obtidos para os testes envolvendo as propostas de paralelização em GPU. Estes resultados fornecem dados para realizar a escolha pela abordagem serial ou paralela, de acordo com a escala utilizada para a rasterização. Na terceira subseção, o algoritmo é testado utilizando os casos de referência da literatura e comparações com outras soluções são realizadas.

4.7.1 Paralelização da geração da montanha de penetração discreta

A paralelização massiva em GPU foi adotada para realizar duas tarefas: a criação da montanha de penetração discreta e a composição do mapa de sobreposição total.

Inicialmente são analisados os resultados do primeiro caso. Na geração da montanha de penetração discreta, três passos são realizados. A primeira etapa consiste em gerar o polígono de obstrução, seguida de sua rasterização e, por fim, a transformada distância é aplicada. A paralelização foi empregada para realizar a transformada distância dos polígonos de obstrução dos casos da literatura. O algoritmo foi implementado em linguagem C++, utilizando a arquitetura CUDA para realizar a programação paralela. Os testes foram conduzidos em um computador Xeon E645, 2.40GHz com 48GB de memória e placa de vídeo GeForce GTX 680, de 2GB de memória.

A escolha do tamanho da grade de rasterização se deu pela limitação da implementação do software desenvolvido que calcula a transformada distância. O algoritmo pode ser modificado a fim de possibilitar maior precisão na rasterização. Entretanto, na prática, foi observado que escalas mais altas prejudicam muito o desempenho do algoritmo de empacotamento.

A [Tabela 7](#) apresenta os tempos computacionais obtidos na execução do algoritmo para os casos da literatura. Em cada caso, as coordenadas originais foram multiplicadas por um fator de escala e a dimensão da grade foi mantida constante (1 unidade). Deste modo, quanto maior a escala, maior a precisão da ferramenta geométrica. A implementação do algoritmo serial foi realizada no software Matlab, utilizando uma função nativa para processar a transformada distância. O ganho de processamento corresponde a relação entre o tempo de execução do algoritmo serial e o do tempo de processamento do algoritmo paralelo. Os resultados destacados em vermelho na tabela mostram que, para alguns casos, o método paralelo obteve desempenho pior em relação ao serial. É possível observar que, na maioria desses casos, a área média dos polígonos de obstrução, dada em número de pixels, é relativamente pequena. É seguro assumir que, nos casos em que as matrizes que representam os itens são muito pequenas, o ganho é baixo e o *overhead* gerado pela paralelização desfavorece a adoção desta implementação. Os casos Jakobs1, Jakobs2 e

Tabela 7 – Resultados da paralelização da transformada distância

Caso	Escala	Área Média ($\times 10^5$)	Tempo de execução (s)		Ganho
			Serial	Paralelo	
Albano	0,8	66,22	76,19	53,90	1,41
Dagli	100	51,30	98,87	70,50	1,40
Dighe1	40	76,76	120,21	70,25	1,71
Dighe2	40	97,18	58,43	33,52	1,74
Fu	100	48,62	336,79	246,25	1,37
Jakobs1	100	8,16	195,22	282,38	0,69
Jakobs2	50	7,80	257,37	375,92	0,68
Mao	1	11,15	80,89	92,44	0,88
Marques	50	45,26	196,03	148,12	1,32
Shapes0	100	31,42	2,87	2,79	1,03
Shapes1	100	30,21	6,16	6,17	1,00
Shapes2	100	6,26	3,21	5,27	0,61
Shirts	100	15,25	16,25	19,56	0,83
Swim	1	45,31	110,57	80,06	1,38
Trousers	50	22,73	81,89	94,39	0,87

Fonte: o autor

Shapes2, que possuem a menor área média dos itens, apresentaram os piores ganhos.

Outro fator que influencia o ganho da implementação paralela é a precisão adotada na rasterização. A fim de verificar o impacto deste parâmetro, o caso Albano foi testado com várias escalas. A Figura 54 mostra que o ganho do algoritmo é elevado em conjunto com a precisão, mesmo que o tempo total de execução aumente.

Figura 54 – Ganhos de processamento da transformada distância para o caso Albano

Fonte: o autor

4.7.2 Paralelização da obtenção do mapa de sobreposição total

As dimensões do mapa de sobreposição total, que é gerado a partir da soma das montanhas de penetração discretas, correspondem aos valores de altura e largura do polígono de posicionamento interno rasterizado. A paralelização de operações de soma de matrizes foi implementada em CUDA e este procedimento foi incluído no algoritmo de empacotamento desenvolvido. Desta forma, os testes foram realizados diretamente no algoritmo de empacotamento, que foi implementado utilizando o compilador Visual Studio 12.0 e o código foi elaborado em C++.

Os testes foram executados em um computador Intel Core i7 870, 2.93Ghz com 16GB de memória e placa de vídeo GeForce GTX 980, de 8GB de memória. Cada execução obedeceu ao tempo limite adotado para os testes de empacotamento, que são descritos na Tabela 9. Por razões de praticidade, as escalas utilizadas também correspondem às maiores adotadas nos testes de resolução do problema de empacotamento de fita. Os resultados são mostrados na Tabela 8. O desempenho é medido pela contagem de iterações do algoritmo de empacotamento, que considera o movimento de todos os itens do leiaute. Desta forma, em uma iteração, o número de mapas gerados é $\sum_{i=1}^n \mathcal{O}_i$, onde n é o número total de itens e $\mathcal{O}_i$ é o total de orientações admitidas para o item i . O limiar apresentado na tabela corresponde ao menor valor da escala que resulta em um ganho maior que 1. Este valor foi obtido a partir da execução do algoritmo considerando várias escalas distintas.

Tabela 8 – Resultados da paralelização da geração do mapa de sobreposição total

Caso	Escala	Iterações por segundo		Ganho	Limiar
		Serial	Paralelo		
Albano	0,1	4,91	21,85	4,45	0,03
Dagli	10	5,19	21,91	4,22	3,26
Dighe1	10	3,19	33,01	10,36	1,71
Dighe2	10	14,83	71,49	4,82	2,29
Fu	25	4,21	12,99	3,08	6,10
Jakobs1	25	6,98	18,12	2,60	11,12
Jakobs2	10	15,73	12,87	0,82	6,56
Mao	0,5	1,33	6,44	4,83	0,10
Marques	20	0,41	2,79	6,76	3,11
Shapes0	25	1,58	7,87	4,98	4,16
Shapes1	25	1,26	8,51	6,74	3,47
Shapes2	25	11,54	28,85	2,50	9,53
Shirts	20	0,70	2,04	2,90	5,32
Swim	0,5	0,05	0,56	10,24	< 0,05
Trousers	10	0,77	3,01	3,92	1,38

Fonte: o autor

Os resultados indicam que, para a maior parte dos casos, a adoção do cálculo paralelo do mapa de sobreposição total é bastante vantajoso. Como o ganho foi medido diretamente na execução do algoritmo de empacotamento, os resultados expressados na

tabela refletem o ganho real obtido quando o algoritmo paralelo proposto é utilizado. Em aproximadamente 60% dos casos, o algoritmo é acelerado de um fator maior que 4.

Figura 55 – Ganhos de processamento para a geração do mapa de sobreposição total (Jakobs2)

Fonte: o autor

Jakobs2 é o único caso em que é desvantajoso o cálculo do mapa na GPU. Entretanto, em testes realizados variando as resoluções, foi possível obter ganho maior utilizando algumas escalas maiores que 6, como pode ser visto no gráfico da Figura 55. É possível verificar que o desempenho varia bastante para escalas maiores que o limiar. Este fenômeno também é verificado em outros casos. Durante a compactação do contêiner, o desempenho varia muito, uma vez que soluções para contêineres de grande comprimento são geradas em um tempo muito menor. Desta forma, é possível que as medições não sejam confiáveis, devido ao menor número de iterações em problemas de alta precisão, o que justifica as variações de desempenho obtidas.

4.7.3 Algoritmo de empacotamento de fita

Nesta segunda proposta deste trabalho, ferramentas rasterizadas foram empregadas para solucionar o problema de empacotamento irregular de fita utilizando uma abordagem de minimização de sobreposição. O algoritmo foi desenvolvido em C++ utilizando o software Visual Studio 2013. Os resultados da Tabela 8 foram analisados a fim de determinar os casos em que a paralelização em GPU é favorável. Ademais, para os resultados desta seção, é considerado que o conjunto de montanhas de penetração discretas já foi determinado.

Os problemas da literatura, que podem ser obtidos no site da ESICUP¹, foram utilizados para a realização dos testes com o objetivo de possibilitar a comparação com

¹ <<http://paginas.fe.up.pt/~esicup/tiki-index.php>>

Tabela 9 – Resultados do algoritmo baseado em métodos rasterizados

Caso	Tempo	Escala	Sobreposição		Multirresolução (sobreposição)			Sem sobreposição	
			Máx.	Méd.	Escala aux.	Máx.	Méd.	Máx.	Méd.
Albano	1200	0,05	87,93	87,21				87,23	86,26
		0,1	88,74	87,39	0,02	87,49	87,14	86,79	86,30
Dagli	1200	5	86,31	85,95				86,31	85,87
		10	86,45	85,92	2	85,72	85,65	86,16	85,60
Dighe1	600	2	100,00	100,00				100,00	100,00
		10	100,00	99,58	2	100,00	100,00	100,00	97,70
Dighe2	600	2	100,00	100,00				100,00	100,00
		10	100,00	99,95	2	100,00	100,00	100,00	99,90
Fu	600	10	90,48	89,15				90,76	89,37
		25	88,95	88,05	5	89,06	88,34	88,73	86,98
Jakobs1	600	10	89,09	88,85				89,09	88,85
		25	88,77	87,16	5	89,09	88,45	89,09	87,56
Jakobs2	600	5	83,19	80,69				80,42	80,35
		10	80,42	79,88	2	80,42	80,42	80,42	79,92
Mao	1200	0,2	84,47	83,11				83,04	82,34
		0,5	84,42	82,25	0,1	83,28	82,21	82,34	81,43
Marques	1200	10	89,14	88,19				88,12	87,51
		20	89,49	87,64	4	88,12	87,15	88,46	86,82
Shapes0	1200	10	67,40	66,32				66,83	66,17
		25	67,04	65,87	5	67,40	66,88	65,97	65,44
Shapes1	1200	10	73,89	72,77				73,89	73,05
		25	73,40	71,30	5	72,81	71,53	73,24	72,14
Shapes2	1200	10	81,51	80,84				80,90	80,42
		25	80,72	80,14	5	81,82	80,96	80,84	80,04
Shirts	1200	10	85,58	84,76				86,82	85,61
		20	85,17	84,03	5	86,54	84,96	86,40	84,24
Swim	1200	0,1	70,96	69,93				70,84	69,94
		0,5	71,27	68,42	0,1	68,67	67,44	70,44	68,82
Trousers	1200	5	89,71	88,85				88,83	88,37
		10	88,79	87,95	2	89,63	88,59	89,71	88,87

Fonte: o autor

outros métodos. Além disso, o conjunto de problemas adotado exibe uma grande variação de propriedades e, desta forma, os resultados podem indicar pontos fortes e fracos do algoritmo. Por fim, os tempos limites estabelecidos para os testes também estão de acordo com outras soluções na literatura que utilizam a minimização de sobreposição. Para todos os testes em que algum valor médio é calculado, o algoritmo foi executado dez vezes.

Para cada problema, duas escalas foram obtidas de modo empírico. O objetivo na escolha destes valores foi a obtenção de duas execuções bastante distintas do ponto de vista de desempenho e eficiência do leiaute final. Um limite superior para a escala foi

estimado de modo a possibilitar um número mínimo de cinco repetições do nível interno do algoritmo. Os demais parâmetros utilizados foram:

- N_{mo} (máximo número de iterações sem aprimoramento no leiaute): 200.
- p_{red} (porcentagem de aumento do comprimento): 1%.
- p_{exp} (porcentagem de redução do comprimento): 3%.

A [Tabela 9](#) apresenta as densidades médias e máximas, dadas em porcentagem, obtidas para três variações do algoritmo. O tempo para cada caso é fornecido em segundos. Foram consideradas duas possibilidades na rasterização dos polígonos de obstrução. A primeira consiste na admissão de sobreposição entre os itens no leiaute final. O valor máximo permitido corresponde à diagonal do quadrado definido por uma unidade da grade. A outra opção foi rejeitar a possibilidade da existência de sobreposição. Neste caso, a detecção pode considerar que itens separados estão se encostando, o que pode prejudicar o resultado. Por fim, a terceira variante da solução considerou a utilização da abordagem com duas resoluções (escalas). Neste caso, foi admitida apenas a rasterização que permite a sobreposição. Uma vez que a busca é realizada com uma precisão auxiliar muito menor que a adotada para o posicionamento, ao considerar a rasterização mais restritiva, o espaço de busca era demasiadamente limitado e testes preliminares do algoritmo geralmente não convergiam para uma solução válida, mesmo para contêineres relativamente grandes. Os melhores leiautes obtidos para os casos com sobreposição e sem sobreposição estão exibidos na [Figura 60](#) e [Figura 61](#), respectivamente.

A primeira análise a ser realizada consiste na comparação dos resultados dos dois métodos que admitem a sobreposição e compreendem as colunas 4-8 da [Tabela 9](#). Ambos utilizam uma modificação para elevar o desempenho, na primeira estratégia é utilizado o cálculo em GPU do mapa de sobreposição total, já no caso de multirresolução, uma rasterização auxiliar é adotada. Considerando apenas testes com maior precisão para cada caso, é possível afirmar que, em geral, os resultados obtidos utilizando duas escalas foram melhores. Como pode ser observado no gráfico da [Figura 56](#), a variante multirresolução obteve eficiências médias melhores ou iguais em 66% dos casos. Entretanto, este valor é de 46% quando consideradas as compactações máximas obtidas.

A variante do algoritmo que não admite sobreposição no leiaute final teve desempenho próximo às soluções que admitem sobreposição, como pode ser verificado no gráfico da [Figura 56](#). Nesta análise, foram considerados, para o caso da rasterização que admite sobreposição, os melhores resultados obtidos para a maior escala adotada e, no caso sem sobreposição foram incluídas todas as escalas testadas para cada caso de modo que o resultado apresentado é o melhor entre as duas resoluções. Desta forma, o valor da sobreposição total dos leiautes é minimizado a fim de possibilitar uma comparação mais adequada. É

Figura 56 – Comparação das soluções das variantes do algoritmo proposto (maior escala)

Fonte: o autor

importante notar que, para os testes sem sobreposição que adotaram escalas menores, o desempenho do algoritmo foi consideravelmente melhor e, conseqüentemente, foi possível realizar uma busca mais abrangente no mesmo intervalo de tempo. Se forem incluídos na comparação os resultados de menor precisão para os algoritmos que admitem sobreposição, é possível observar que existe uma significativa diferença positiva nas densidades obtidas utilizando esta abordagem, como pode ser observado no gráfico da [Figura 57](#).

Devido à possibilidade da obtenção de leiautes inválidos, não é adequado comparar os resultados da variante que admite sobreposição com as propostas na literatura que empregam a representação poligonal e, conseqüentemente, não admitem sobreposição entre itens. Tampouco a comparação com a primeira proposta deste trabalho deve considerar

Figura 57 – Comparação das soluções das variantes do algoritmo proposto (menor escala)

Fonte: o autor

estas soluções. Desta forma, apenas os resultados das últimas duas colunas são considerados na comparação com outras soluções.

A [Tabela 10](#) e a [Figura 58](#) apresentam um estudo das densidades médias geradas por propostas de minimização de sobreposição. A [Tabela 11](#) e a [Figura 59](#) exibem as compactações máximas obtidas, incluindo as melhores soluções da literatura. As soluções consideradas foram: Guided Cuckoo Search (GCS) ([ELKERAN, 2013](#)), 2DNest ([EGE-BLAD; NIELSEN; ODGAARD, 2007](#)), ILSQN ([IMAMICHI; YAGIURA; NAGAMOCHI, 2009](#)) e Extended Local Search (ELS) ([LEUNG; LIN; ZHANG, 2012](#)). Os tempos de execução são similares aos adotados nesta proposta e são apresentados na [Tabela 9](#), com exceção do tempo de execução utilizado pelo algoritmo 2DNest para a obtenção da densidade

Tabela 10 – Densidades médias (em porcentagem) obtidas por soluções na literatura comparadas às obtidas pelo algoritmo baseado em métodos rasterizados

	Algoritmo proposto	2DNest	ILSQN	ELS	GCS
Albano	86,30	86,96	87,14	87,38	87,47
Dagli	85,87	85,31	85,80	86,27	87,06
Dighe1	100,00	93,93	90,49	91,61	100,00
Dighe2	100,00	93,11	84,21	100,00	100,00
Fu	89,37	90,93	87,57	90,00	90,68
Jakobs1	88,85	88,90	84,78	88,35	88,90
Jakobs2	80,35	80,28	80,50	80,97	81,14
Mao	82,34	82,67	81,31	82,57	82,93
Marques	87,51	88,73	86,81	88,32	89,40
Shapes0	66,17	65,42	66,49	66,85	67,26
Shapes1	73,05	71,74	72,83	74,24	73,79
Shapes2	80,42	79,89	81,72	82,55	82,40
Shirts	85,61	85,73	88,12	87,20	87,59
Swim	69,94	70,27	74,62	74,10	74,49
Trousers	88,87	89,29	88,69	88,29	89,02

Fonte: o autor

máxima, que foi de 6 horas. Com relação às densidades médias, o algoritmo proposto obteve resultados bastante competitivos, na maior parte dos casos alcançando valores com diferenças menores que 1% na compactação. O mesmo pode ser observado, com diferença um pouco maior, nas densidades máxima. Em ambos os casos, a eficiência do algoritmo GCS é consideravelmente maior.

Por fim, é realizada a comparação entre as duas propostas deste trabalho na [Tabela 12](#). Os resultados mostram que, apesar de adotar um limite de tempo bem menor, na maior parte dos casos, a proposta rasterizada apresentou melhor compactação.

Tabela 11 – Densidades máximas (em porcentagem) obtidas por soluções na literatura comparadas às obtidas pelo algoritmo baseado em métodos rasterizados

	Algoritmo proposto	2DNest	ILSQN	ELS	GCS
Albano	87,23	87,88	88,16	88,48	89,58
Dagli	86,31	87,05	87,40	88,11	89,51
Dighe1	100,00	99,84	99,89	100,00	100,00
Dighe2	100,00	93,02	99,99	100,00	100,00
Fu	90,76	92,03	90,67	91,94	92,41
Jakobs1	89,09	89,03	86,89	89,09 ¹	89,09 ²
Jakobs2	80,42	81,07	82,51	83,92	87,73
Mao	83,04	85,15	83,44	84,33	85,44
Marques	88,46	89,82	89,03	89,73	90,59
Shapes0	66,61	66,42	68,44	67,63	68,79
Shapes1	73,89	73,23	73,84	75,29	76,73
Shapes2	80,90	81,59	84,25	84,23	84,84
Shirts	86,82	87,38	88,78	88,40	88,96
Swim	70,84	72,49	75,29	75,43	75,94
Trousers	88,83	90,46	89,79	89,63	91,00

¹ Em [Leung, Lin e Zhang \(2012\)](#), o autor apresenta 89,10% como valor de compactação máxima para Jakobs2, porém é possível inferir a partir do leiaute que o valor real é 89,09%.

² Ver nota anterior.

Fonte: o autor

Tabela 12 – Comparação das duas propostas deste trabalho

	Proposta com encaixe perfeito			Proposta rasterizada		
	Máximo (%)	Média (%)	Tempo (s)	Máximo (%)	Média (%)	Tempo (s)
Albano	86,66	86,05	7200	87,40	86,86	1200
Dagli	85,23	83,89	7200	86,31	85,87	1200
Dighe1	100,00	99,96	7200	100,00	100,00	600
Dighe2	89,61	89,54	7200	100,00	100,00	600
Fu	91,84	91,11	7200	90,76	89,37	600
Jakobs1	89,07	88,82	7200	89,09	88,85	600
Jakobs2	80,42	80,10	7200	80,42	80,35	600
Mao	81,90	79,80	7200	84,23	82,78	1200
Marques	88,91	87,71	7200	88,46	87,51	1200
Shapes0	66,50	65,71	7200	66,61	66,17	1200
Shapes1	70,82	69,46	7200	73,89	73,05	1200
Shapes2	82,03	80,45	7200	80,90	80,42	1200
Shirts	84,48	83,13	21600	86,82	85,61	1200
Swim	67,65	65,32	21600	71,32	70,34	1200
Trousers	87,57	86,07	21600	88,83	88,87	1200

Fonte: o autor

Figura 58 – Comparação dos resultados com as soluções da literatura (densidades médias)

Fonte: o autor

Figura 59 – Comparação dos resultados com as soluções da literatura (densidades máximas)

Fonte: o autor

Figura 60 – Leiautes mais compactos obtidos através do algoritmo baseado em métodos rasterizados (com sobreposição)

Fonte: o autor

Figura 61 – Leiautes mais compactos obtidos através do algoritmo baseado em métodos rasterizados (sem sobreposição)

Fonte: o autor

5 Discussão dos resultados

Na primeira proposta deste trabalho, a região degenerada foi utilizada a fim de elevar a ocorrência de posições de encaixe. Com o objetivo de aplicar esta técnica, heurísticas construtivas foram propostas, sendo que a sobreposição não foi admitida durante a solução do problema. Os resultados indicaram que o espaço de busca era demasiadamente desconexo, uma vez que o algoritmo de recozimento simulado exigiu muito tempo para convergir. Desta forma, os resultados obtidos não foram bastante eficientes, o que foi comprovado pela comparação com soluções da literatura. Uma tentativa de ampliar o espaço de soluções consistiu na remoção da heurística de posicionamento, transferindo o controle da posição para o recozimento simulado. Entretanto, os resultados não indicaram nenhuma melhora.

A segunda proposta, baseada em métodos rasterizados, foi desenvolvida de modo a possibilitar um espaço de soluções conexo. Assim, foi adotada a abordagem de minimização de sobreposição. Neste caso, os resultados obtidos foram competitivos com os da literatura e, em relação com a primeira proposta, as compactações obtidas foram consideravelmente maiores. O maior problema verificado, que é comum a propostas rasterizadas, foi a definição da precisão. Idealmente, a maior precisão possível é desejada. No entanto, dois problemas são encontrados quando uma escala de rasterização muito alta é adotada: o alto requisito de memória e a redução do desempenho. A partir dos testes, foi possível observar que o principal obstáculo encontrado foi o baixo desempenho do algoritmo quando uma alta resolução foi adotada. Isto posto, foram propostas técnicas para aprimorar o desempenho, de modo a possibilitar a obtenção de soluções compactas com maior precisão.

Duas estratégias foram propostas para reduzir o tempo computacional do algoritmo: a paralelização em GPU e a utilização de duas resoluções. Inicialmente, foi considerada a possibilidade da combinação das duas propostas. Entretanto, testes indicaram que a paralelização não era eficiente quando escalas de rasterização baixas eram adotadas. Por outro lado, o algoritmo multirresolução só se mostrou eficiente quando uma das escalas era consideravelmente reduzida, de modo que o espaço de movimentação dos itens era bastante simplificado. Por este motivo, testes foram realizados utilizando apenas uma das técnicas e os resultados mostraram que, embora ambas reduziram o tempo computacional, nenhuma das propostas pôde ser considerada definitivamente mais eficiente.

Outro problema encontrado foi o impacto da qualidade do comprimento inicial do contêiner. Neste trabalho, a solução inicial foi obtida aplicando a heurística inferior esquerda a partir de uma sequência aleatória de posicionamento. O contêiner é então dimensionado de forma a minimizar o seu comprimento. Foi detectado que este valor

variava bastante, o que comprometia a qualidade da solução final. Isto ocorreu devido à necessidade do algoritmo solucionar vários problemas de eliminação de sobreposição, um para cada comprimento. Desta forma, quanto maior for o comprimento inicial do contêiner, mais vezes deve ser resolvido o problema de eliminação da sobreposição para atingir um leiaute com a mesma densidade.

Por fim, foram encontradas dificuldades na escolha da ferramenta rasterizada para detectar sobreposição. A rasterização mapeia as configurações relativas inválidas para dois itens. Uma vez que não é possível obter todas as posições em que os itens se encostam, que são as mais importantes para o empacotamento, foi considerada a possibilidade de admitir uma pequena sobreposição entre itens, determinada a partir da escala de rasterização escolhida. Esta alternativa obteve melhores resultados, especialmente com escalas menores, como era esperado. No entanto, não é apropriada a comparação destes resultados com outras propostas, uma vez que, nestes estudos, geralmente não é admitida sobreposição. Uma solução para possibilitar a comparação destes resultados consiste na aplicação de um método de separação e compactação, que considera a representação poligonal, garantindo a validade do leiaute. Ademais, é possível obter uma compactação ainda maior com a aplicação destas técnicas, uma vez que mais possibilidades de movimentação para os itens são admitidas.

6 Conclusão

Neste trabalho, foram propostos dois algoritmos para a solução do problema de empacotamento irregular de fita. O primeiro visou maximizar os posicionamentos de encaixe, que podem ser importantes para a compactação do leiaute. Nesta solução, itens foram inseridos aos pares no contêiner e um algoritmo de recozimento simulado foi adotado para guiar a busca no espaço de soluções. Os resultados obtidos não apresentaram uma eficiência boa quando comparada a outros métodos e o tempo de execução foi relativamente elevado, apesar do espaço de busca adotado ser bem limitado. Uma hipótese para explicar este problema é a baixa conectividade do espaço de soluções. Desta forma, devido à representação adotada para a solução, pequenas modificações ocasionaram grandes perturbações na solução e no valor da função objetivo. Além disso, foi verificado nos testes que nem todas as posições de encaixe favorecem a compactação do leiaute e não foi possível obter um algoritmo para escolher os melhores encaixes.

Na segunda abordagem, foi escolhido o desenvolvimento de um método baseado na minimização da sobreposição. Nesta solução, os itens podem se sobrepor durante a busca pela solução final, que deve ser válida. A fim de facilitar a busca na movimentação dos itens, foi admitida uma grade de posicionamento, que reduz o espaço de busca para a translação dos itens. Esta discretização possibilitou a utilização de técnicas rasterizadas para melhorar o desempenho do algoritmo. Testes foram realizados e foram obtidas soluções competitivas quando comparadas a outras soluções, embora em nenhum caso o leiaute mais compacto da literatura foi obtido. Apesar do desempenho satisfatório, foram identificados alguns obstáculos para a obtenção de leiautes ainda mais compactos. O primeiro foi a baixa qualidade das soluções iniciais obtidas e o segundo foi a incapacidade de utilizar os resultados que permitiam sobreposição. Deste modo, o desenvolvimento de um método mais eficiente para a obtenção do leiaute inicial válido e a aplicação de técnicas matemáticas de separação e compactação são deixados como trabalhos futuros. Os resultados indicam que existe potencial para melhoria caso estes métodos sejam aplicados.

Referências

- AGARWAL, P. K.; FLATO, E.; HALPERIN, D. Polygon decomposition for efficient construction of minkowski sums. *Computational Geometry*, v. 21, n. 1–2, p. 39 – 61, 2002. Sixteenth European Workshop on Computational Geometry - EUROCG-2000. Citado na página [23](#).
- ALBANO, A.; SAPUPPO, G. Optimal allocation of two-dimensional irregular shapes using heuristic search methods. *Systems, Man and Cybernetics, IEEE Transactions on*, v. 10, n. 5, p. 242–248, Maio 1980. Citado na página [30](#).
- ALVAREZ-VALDES, R.; MARTINEZ, A.; TAMARIT, J. A branch & bound algorithm for cutting and packing irregularly shaped pieces. *International Journal of Production Economics*, v. 145, n. 2, p. 463 – 477, 2013. Citado 2 vezes nas páginas [27](#) e [29](#).
- ART, R. C. An approach to the two-dimensional irregular cutting stock problem. *Technical Report 36.008, IBM Cambridge Centre*, 1966. Citado na página [21](#).
- BABU, A. R.; BABU, N. R. A generic approach for nesting of 2D parts in 2D sheets using genetic and heuristic algorithms. *Computer Aided Design*, v. 33, p. 879–891, 2001. Citado 3 vezes nas páginas [21](#), [27](#) e [30](#).
- BENNELL, J. A.; DOWSLAND, K. A. A tabu thresholding implementation for the irregular stock cutting problem. *International Journal of Production Research*, v. 37, n. 18, p. 4259–4275, 1999. Citado 3 vezes nas páginas [26](#), [27](#) e [32](#).
- BENNELL, J. A.; DOWSLAND, K. A. Hybridising tabu search with optimisation techniques for irregular stock cutting. *Management Science, INFORMS*, v. 47, n. 8, p. 1160–1172, 2001. Citado na página [32](#).
- BENNELL, J. A.; DOWSLAND, K. A.; DOWSLAND, W. B. The irregular cutting-stock problem — a new procedure for deriving the no-fit polygon. *Computers & Operations Research*, v. 28, n. 3, p. 271 – 287, 2001. Citado na página [23](#).
- 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. Citado 2 vezes nas páginas [19](#) e [21](#).
- BENNELL, J. A.; OLIVEIRA, J. F. A tutorial in irregular shape packing problems. *The Journal of the Operational Research Society*, Palgrave Macmillan Journals on behalf of the Operational Research Society, v. 60, p. s93–s105, 2009. Citado 2 vezes nas páginas [26](#) e [30](#).
- BENNELL, J. A.; SONG, X. A comprehensive and robust procedure for obtaining the nofit polygon using Minkowski sums. *Computers & Operations Research*, v. 35, n. 1, p. 267 – 281, 2008. Citado na página [23](#).
- BENNELL, J. A.; SONG, X. A beam search implementation for the irregular shape packing problem. *Journal of Heuristics*, Springer Netherlands, v. 16, p. 167–188, 2010. Citado 2 vezes nas páginas [31](#) e [63](#).

- BLAZEWICZ, J.; HAWRYLUK, P.; WALKOWIAK, R. Using a tabu search approach for solving the two-dimensional irregular cutting problem. *Annals of Operations Research*, Baltzer Science Publishers, Baarn/Kluwer Academic Publishers, v. 41, n. 4, p. 313–325, 1993. Citado 2 vezes nas páginas 27 e 30.
- BURKE, E. et al. A new bottom-left-fill heuristic algorithm for the two-dimensional irregular packing problem. *Operations Research*, v. 54, n. 3, p. 587–601, 2006. Citado na página 30.
- BURKE, E. et al. Complete and robust no-fit polygon generation for the irregular stock cutting problem. *European Journal of Operational Research*, v. 179, n. 1, p. 27 – 49, 2007. Citado na página 23.
- BURKE, E. K. et al. Irregular packing using the line and arc no-fit polygon. *Operations Research*, v. 58, n. 4-part-1, p. 948–970, 2010. Citado na página 25.
- CARRAVILLA, M. A.; RIBEIRO, C.; OLIVEIRA, J. F. Solving nesting problems with non-convex polygons by constraint logic programming. *International Transactions in Operational Research*, Blackwell Publishing Ltd, v. 10, n. 6, p. 651–663, 2003. Citado na página 29.
- CHERNOV, N. et al. Phi-functions for 2d objects formed by line segments and circular arcs. *Advances in Operations Research*, v. 2012, 2012. Citado na página 23.
- DANIELSSON, P.-E. Euclidean distance mapping. *Computer Graphics and Image Processing*, v. 14, n. 3, p. 227 – 248, 1980. Citado 2 vezes nas páginas 88 e 89.
- DYCKHOFF, H. A typology of cutting and packing problems. *European Journal of Operational Research*, v. 44, n. 2, p. 145 – 159, 1990. Cutting and Packing. Citado na página 16.
- 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. Citado 4 vezes nas páginas 32, 67, 70 e 99.
- ELKERAN, A. A new approach for sheet nesting problem using guided cuckoo search and pairwise clustering. *European Journal of Operational Research*, v. 231, n. 3, p. 757 – 769, 2013. Citado 7 vezes nas páginas 32, 62, 64, 67, 70, 81 e 99.
- FAROE, O.; PISINGER, D.; ZACHARIASEN, M. Guided local search for the three-dimensional bin-packing problem. *INFORMS Journal on Computing*, v. 15, n. 3, p. 267–283, 2003. Citado na página 32.
- FISCHETTI, M.; LUZZI, I. Mixed-integer programming models for nesting problems. *Journal of Heuristics*, Springer US, v. 15, n. 3, p. 201–226, 2009. Citado na página 29.
- FOWLER, R. J.; PATERSON, M.; TANIMOTO, S. L. Optimal packing and covering in the plane are NP-complete. *Information Processing Letters*, v. 12, n. 3, p. 133–137, 1981. Citado na página 26.
- GOMES, A. M.; OLIVEIRA, J. F. A 2-exchange heuristic for nesting problems. *European Journal of Operational Research*, v. 141, p. 359–370, 2002. Citado na página 31.

- 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. Citado 4 vezes nas páginas 27, 28, 32 e 60.
- HECKMANN, R.; LENGAUER, T. A simulated annealing approach to the nesting problem in the textile manufacturing industry. *Annals of Operations Research*, Baltzer Science Publishers, Baarn/Kluwer Academic Publishers, v. 57, n. 1, p. 103–133, 1995. Citado 2 vezes nas páginas 28 e 31.
- HOFFMANN, C. M. *Geometric and Solid Modeling: An Introduction*. San Francisco, CA, USA: Morgan Kaufmann Publishers, 1989. Citado na página 45.
- IMAMICHI, T.; YAGIURA, M.; NAGAMOCCHI, H. An iterated local search algorithm based on nonlinear programming for the irregular strip packing problem. *Discrete Optimization*, v. 6, p. 345–361, 2009. Citado 7 vezes nas páginas 27, 28, 32, 67, 70, 84 e 99.
- JAKOBS, S. On genetic algorithms for the packing of polygons. *European Journal of Operational Research*, v. 88, n. 1, p. 165 – 181, 1996. Citado 2 vezes nas páginas 27 e 30.
- KIRKPATRICK, S.; GELLAT, C. D.; VECCHI, M. P. Optimization by simulated annealing. *Science*, v. 220, p. 671–680, 1983. Citado na página 58.
- LEUNG, S. C.; LIN, Y.; ZHANG, D. Extended local search algorithm based on nonlinear programming for two-dimensional irregular strip packing problem. *Computers & Operations Research*, v. 39, n. 3, p. 678 – 686, 2012. Citado 4 vezes nas páginas 32, 67, 99 e 101.
- LI, Z.; MILENKOVIC, V. Compaction and separation algorithms for non-convex polygons and their applications. *European Journal of Operational Research*, v. 84, n. 3, p. 539 – 561, 1995. Citado na página 28.
- MAHADEVAN, A. *Optimization in computer-aided pattern packing (marking, envelopes)*. Tese (Doutorado) — North Carolina State University, 1984. AAI8507009. Citado 2 vezes nas páginas 21 e 22.
- MARTINS, T. C.; TSUZUKI, M. S. G. Simulated annealing applied to the rotational polygon packing. *Proceedings of the 12th IFAC Symposium Information Control Problems in Manufacturing*, p. 475–480, 2006. Citado 3 vezes nas páginas 35, 36 e 40.
- 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 13th IFAC Symposium Information Control Problems in Manufacturing*, 2009a. Citado na página 36.
- 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, 2009b. Citado na página 36.
- 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. Citado na página 36.

- 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. Citado na página 31.
- SATO, A. K.; MARTINS, T. C.; TSUZUKI, M. S. G. An algorithm for the strip packing problem using collision free region and exact fitting placement. *Computer-Aided Design*, v. 44, n. 8, p. 766 – 777, 2012. Citado 6 vezes nas páginas 14, 27, 36, 56, 58 e 62.
- SATO, A. K.; MARTINS, T. C.; TSUZUKI, M. S. G. Collision free region determination by modified polygonal boolean operations. *Computer-Aided Design*, v. 45, n. 7, p. 1029 – 1041, 2013. Citado 4 vezes nas páginas 33, 36, 44 e 45.
- SCHNEIDER, J.; KRAUS, M.; WESTERMANN, R. GPU-based real-time discrete euclidean distance transforms with precise error bounds. *International Conference on Computer Vision Theory and Applications (VISAPP)*, p. 435 – 442, 2009. Citado 2 vezes nas páginas 89 e 91.
- SEGENREICH, S. A.; BRAGA, L. M. P. F. Optimal nesting of general plane figures: A monte carlo heuristical approach. *Computers & Graphics*, v. 10, n. 3, p. 229 – 237, 1986. Citado na página 21.
- STOYAN, Y. et al. Phi-functions for primary 2d-objects. *Studia Informatica Universalis*, v. 2, n. 1, p. 1–32, 2001. Citado na página 23.
- TOLEDO, F. M. et al. The dotted-board model: A new MIP model for nesting irregular shapes. *International Journal of Production Economics*, v. 145, n. 2, p. 478 – 487, 2013. Citado 3 vezes nas páginas 26, 27 e 29.
- UMETANI, S. et al. Solving the irregular strip packing problem via guided local search for overlap minimization. *International Transactions in Operational Research*, Blackwell Publishing Ltd, v. 16, n. 6, p. 661–683, 2009. Citado 3 vezes nas páginas 32, 67 e 81.
- VEERAMANI, D.; GAU, Y.-S. Selection of an optimal set of cutting-tool sizes for 2d pocket machining. *Computer-Aided Design*, v. 29, n. 12, p. 869 – 877, 1997. Citado na página 74.
- 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. Citado 3 vezes nas páginas 16, 17 e 19.
- WHITWELL, G. *Novel Heuristic and Metaheuristic Approaches to Cutting and Packing*. Tese (Doutorado) — School of Computer Science and Information Technology, University of Nottingham, 2005. Citado na página 23.