

**UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"  
FACULDADE DE ENGENHARIA  
CAMPUS DE ILHA SOLTEIRA**

**ELIANE VENDRAMINI DE OLIVEIRA**

**OTIMIZAÇÃO DO PROBLEMA DE CORTE BIDIMENSIONAL NÃO  
GUILHOTINADO USANDO META-HEURÍSTICAS ESPECIALIZADAS**

Ilha Solteira  
2018



UNIVERSIDADE ESTADUAL PAULISTA  
"JÚLIO DE MESQUITA FILHO"  
Campus de Ilha Solteira

**ELIANE VENDRAMINI DE OLIVEIRA**

**OTIMIZAÇÃO DO PROBLEMA DE CORTE BIDIMENSIONAL NÃO  
GUILHOTINADO USANDO META-HEURÍSTICAS ESPECIALIZADAS**

Tese apresentada à Faculdade de Engenharia de Ilha Solteira – UNESP, como parte dos requisitos para obtenção do título de Doutora em Engenharia Elétrica.

Área do Conhecimento: Automação.

Orientador: Prof. Dr. RUBÉN AUGUSTO  
ROMERO LÁZARO

Ilha Solteira  
2018

FICHA CATALOGRÁFICA

Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

Oliveira, Eliane Vendramini de.  
O482o Otimização do problema de corte bidimensional não guilhotinado usando meta-heurísticas especializadas / Eliane Vendramini de Oliveira. -- Ilha Solteira: [s.n.], 2018  
155 f. : il.

Tese (doutorado) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira. Área de conhecimento: Automação, 2018

Orientador: Rubén Augusto Romero Lázaro  
Inclui bibliografia

1. Problema de corte bidimensional não guilhotinado. 2. Meta-heurísticas. 3. Algoritmo genético. 4. RVNS.

*Ruiane da Silva Santos*  
Ruiane da Silva Santos

Supervisor Técnico de Serviço  
Seção Técnica de Referência, Atendimento ao usuário e Documentação  
Departamento Técnico de Biblioteca e Documentação  
CRB2/908

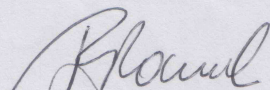
**CERTIFICADO DE APROVAÇÃO**

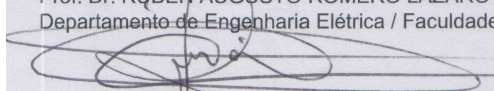
TÍTULO DA TESE: Otimização do problema de corte bidimensional não guilhotinado usando meta-heurística especializada

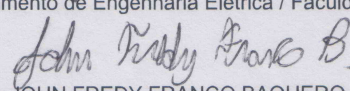
**AUTORA: ELIANE VENDRAMINI DE OLIVEIRA**

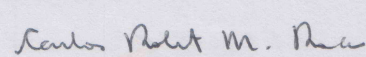
**ORIENTADOR: RUBEN AUGUSTO ROMERO LAZARO**

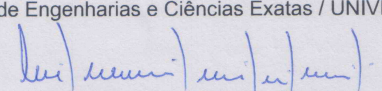
Aprovada como parte das exigências para obtenção do Título de Doutora em ENGENHARIA ELÉTRICA, área: AUTOMAÇÃO pela Comissão Examinadora:

  
Prof. Dr. RUBEN AUGUSTO ROMERO LAZARO  
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira

  
Prof. Dr. JOSE ROBERTO SANCHES MANTOVANI  
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira

  
Prof. Dr. JOHN FREDY FRANCO BAQUERO  
Departamento de Engenharia Elétrica / Universidade Estadual Paulista Júlio de Mesquita Filho, Câmpus Experimental Rosana

  
Prof. Dr. CARLOS ROBERTO MENDONÇA DA ROCHA  
Centro de Engenharias e Ciências Exatas / UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ

  
Prof. Dr. LUIS GUSTAVO WESZ DA SILVA  
Departamento de Industrias / Instituto Federal de Educação, Ciência e Tecnologia de Goiás

Ilha Solteira, 25 de maio de 2018

*Dedico este trabalho ao meu esposo e amigo, Márcio, pelo companheirismo e pela compreensão nos momentos em que me ausentei para me dedicar à construção deste sonho, sempre me apoiando e acreditando em mim. Dedico este trabalho também à minha querida filha, Ana Luisa. Foi pensando em vocês que escrevi cada página desta tese. Amo muito vocês.*

## **AGRADECIMENTOS**

A Deus, primeiramente, por ter me concedido saúde física e mental para chegar até aqui. Agradeço ao meu esposo, Marcio, por ter aceitado participar comigo desta jornada, não deixando que minha ausência, em certos momentos, afetasse a tranquilidade do nosso lar. Agradeço também à minha filha, Ana Luisa, que, mesmo sendo tão pequena, sempre esteve comigo, por isso ela foi lembrada em cada página escrita.

Agradeço à minha família que sempre me apoiou, dando-me forças para continuar e não desistir: meu pai, Rosalino; minha mãe, Lucilei; minhas irmãs, Camile e Daniele.

Agradeço ao meu orientador, professor Doutor Rubén Romero, por estar sempre disponível a me ajudar e a me orientar, por ter a sensibilidade de entender meu tempo e minha dedicação a este trabalho. Certamente, ele foi e sempre será um exemplo a ser seguido.

Agradeço às minhas amigas, Patrícia Fernanda e Monara, que tornaram a caminhada até aqui mais suave.

Agradeço a todos os professores pelos ensinamentos passados e a todos os colegas pelos bons momentos de convívio.

Agradeço à FAPESP pelo apoio financeiro (processo nº 2015/21972-6).

Agradeço, em especial, ao Centro Paula Sousa e à UNESP pela parceria firmada, proporcionando aos professores das ETECs e FATECs darem continuidade à formação profissional.

“Tudo posso naquele que me fortalece”  
(BÍBLIA, FP 4:13)

## RESUMO

O Problema de Corte Bidimensional não guilhotinado tem sua aplicação prática quando comparado a problemas de indústrias que trabalham com aço, madeira, vidro, entre outros materiais, os quais necessitam de um padrão de corte que lhes proporcione maior lucro entre as peças cortadas, usando-se como técnica de corte o laser, e não a guilhotina, por isso existem diversas propostas para a resolução desse problema. Em particular, as propostas de solução utilizando-se meta-heurísticas foram o foco desta pesquisa. Vários trabalhos relevantes nessa área foram analisados, servindo de base para que esta tese trouxesse contribuições para a resolução do problema. A pesquisa sobre o problema permitiu que se apresentasse uma nova forma de representação da proposta de solução para o problema de corte bidimensional não guilhotinado. Outro resultado importante que se apresenta neste trabalho foi o desenvolvimento de duas meta-heurísticas especializadas na resolução do problema de corte bidimensional não guilhotinado. A primeira delas é o algoritmo genético de chaves aleatórias viciadas, e a segunda meta-heurística implementada foi RVNS. Foram realizados vários testes, utilizando-se instâncias conhecidas na literatura especializada, e os resultados encontrados pelas meta-heurísticas algoritmo genético e RVNS propostas pela autora foram de boa qualidade, principalmente se comparados com os resultados já conhecidos na literatura. Os resultados obtidos com o algoritmo genético especializado, em muitos casos, foram iguais aos encontrados na literatura, e em dois casos de testes apresentaram-se superiores, contribuindo novamente para a área especializada no problema. Outro comparativo de resultados realizados pela autora está relacionado aos resultados obtidos pelas meta-heurísticas especialistas, propostas nesta tese, aos resultados encontrados utilizando-se o software AMPL para modelagem matemática em conjunto com o solver CPLEX. Nesse caso, novamente as meta-heurísticas algoritmo genético e RVNS apresentaram resultados iguais ou muito próximos do ótimo encontrado pelo modelo matemático. Os algoritmos desenvolvidos pela autora, além de resolverem o problema de corte bidimensional não guilhotinado, apresentaram bons resultados, visto que promoveram melhorias em relação ao que já existe na literatura. Os algoritmos foram escritos na linguagem de programação Fortran. Foram utilizados casos de teste de pequeno, médio e grande número de peças. Concluiu-se que o problema de corte bidimensional não guilhotinado é complexo e apresenta diversas variantes, sendo que as meta-heurísticas implementadas, neste trabalho, atendem a essa demanda com eficiência. Evidências empíricas mostram que esses algoritmos podem ser apropriados para solucionar instâncias associadas a situações reais.

**Palavras-chave:** Problema de corte bidimensional não guilhotinado. Meta-heurísticas. Algoritmo genético. RVNS.



## ABSTRACT

The two-dimensional non-guillotine cutting problem has its practical application when compared to problems in industries that work with steel, wood, glass, among other materials, which require a cut pattern that provides more profit among the cut pieces, using laser as a cut technique, not the guillotine. Thus, there are several potential answers for this question. In particular, the potential solutions using metaheuristics were the focus of this research. Several relevant papers in this area were analyzed, forming a base so that this dissertation can bring solutions for the problem. The research about this issue allowed us to present a new form of representation of the proposal of solution for the two-dimensional non-guillotine problem. Another important result presented in this paper is the development of two metaheuristics specialized in the resolution of the two-dimensional non-guillotine problem. The first is the biased random-key genetic algorithm. The second metaheuristic was the RVNS. Several tests were performed, using methods well-known in the specialized literature, and the results found by the metaheuristics genetic algorithm and the RVNS suggested by the author were of good quality, mainly if compared to the results already known in the literature. The results obtained by the specialized genetic algorithm, in many cases, were equal to the ones found in the literature, and, in two tests, they were superior, once more contributing to the specialized field of the problem. Another comparison between the results performed by the author is related to the outcomes obtained by the specialized metaheuristics, suggested in this dissertation, and the ones found using the AMPL software to the mathematical modeling together with the CPLEX solver. In this case, once more, the genetic algorithm and RVNS metaheuristics presented resulted identical or very similar to the optimum one found by the mathematical model. The algorithms developed by the author not just solved the two-dimensional non-guillotine cutting problem, but present good results, given that they promoted improvements, comparing to what already exists in the literature. The algorithms were written in the Fortran programming language. Small, medium and big number of pieces' case-tests were performed. The conclusion was that the two-dimensional non-guillotine cutting problem is complex and presents several variants. However, the metaheuristics implemented by this research efficiently meet this demand. Empirical evidences show that these algorithms can be used to solve issues associated with real situations.

**Keywords:** Two-dimensional non-guillotine cutting problem. Metaheuristics. Genetic algorithm. RVNS.

## LISTA DE FIGURAS

<b>Figura 1</b> - Exemplo de Corte Guilhotinado .....	18
<b>Figura 2</b> - Exemplo de Diferentes Estilos de Padrões de Cortes .....	18
<b>Figura 3</b> - Exemplo de Corte Não Guilhotinado .....	19
<b>Figura 4</b> - Exemplo do Corte de Placa Retangular Maior em Peças Retangulares Menores. (a) Padrão de Corte de Peças Menores em uma Placa Maior. (b) Peças Resultantes do Corte da Placa. (c) Produto Finalizado pela Indústria. ....	20
<b>Figura 5</b> - Exemplo de uma Instância Usada para Testar o Problema de Corte Bidimensional Não Guilhotinado. (a) Instância de Beasley (1985). (b) Irrestrito com Otimização: 268. (c) Restrito com Otimização: 247. (d) Duplamente Restrito com Otimização: 220. ....	23
<b>Figura 6</b> - Visão Geral dos Tipos de Problemas Relacionados ao Corte e ao Empacotamento .....	30
<b>Figura 7</b> - Métodos de Resolução de Problemas de Otimização .....	31
<b>Figura 8</b> - Exemplo de um <i>Tour</i> com 10 Cidades para o Problema do Caixeiro-Viajante ..	40
<b>Figura 9</b> - Vetor Ordenado de Corte das Peças na Placa por Lai e Chan (1997) .....	41
<b>Figura 10</b> - O Padrão de Corte é Ilustrado pela Ordem de Corte [a1, a2, a3, a4] .....	44
<b>Figura 11</b> - O Padrão de Corte é Ilustrado pela Ordem de Corte [a1, a3, a2, a4] .....	44
<b>Figura 12</b> - Processo de Diferença na Geração do Intervalo .....	45
<b>Figura 13</b> - Processo de Eliminação dos Intervalos Gerados .....	47
<b>Figura 14</b> - Seleção de Retângulo Livre .....	50
<b>Figura 15</b> - Estrutura de Vizinhança. (a) Se o Valor Âncora é 5, Quaisquer Dois Elementos Entre $a_2$ e $a_6$ Podem Ser Selecionados para a Troca. (b) Se o Valor Âncora é 2, Apenas os Elementos Entre $a_5$ e $a_6$ Podem Ser Selecionados para a Troca. ....	53
<b>Figura 16</b> - Mesclar Dois Retângulos Livres.....	59
<b>Figura 17</b> - Retirada de Blocos. (a) Seleção, (b) Retirada, (c) Transição para o Canto e (d) Preencher .....	61
<b>Figura 18</b> - Retângulos Livres na Transição de Retirada. (a) Passo 1, (b) Passo 2, (c) Mesclar e (d) Preencher.....	62
<b>Figura 19</b> - Selecionando a Posição do Novo Bloco. (a) Maior Área Comum e (b) Posições Possíveis .....	63
<b>Figura 20</b> - Inserção de Blocos. (a) Inicial, (b) Inserir, (c) Eliminar Sobreposição e (d) Mesclar e Preencher .....	64
<b>Figura 21</b> - Representação da Proposta de Solução Codificada por Gonçalves e Resende (2013).....	66
<b>Figura 22</b> - Decodificação da Sequência de Peças Cortadas.....	67
<b>Figura 23</b> - Diferença Entre as Três Representações de Solução para o Problema.....	75
<b>Figura 24</b> - Fluxograma do Funcionamento dos Algoritmos Genéticos Tradicionais .....	81

<b>Figura 25</b> - Modelo da Roleta do Método de Seleção Proporcional. ....	84
<b>Figura 26</b> - Decodificador que Procura a Solução no Espaço de Busca Original .....	91
<b>Figura 27</b> - Diagrama de Fluxo do Algoritmo BRKGA.....	93
<b>Figura 28</b> - Proposta de Solução Inicial Codificada Utilizada pelo Algoritmo Genético de Chaves Aleatórias Viciadas Especializado .....	94
<b>Figura 29</b> - Escaneamento de Retângulos Livres. (a) Primeiro Retângulo Livre Escaneado, (b) Segundo Retângulo Livre Escaneado .....	99
<b>Figura 30</b> - Escaneamento de Retângulos Livres Depois do Corte da Segunda Peça. (a) Primeiro Retângulo Livre Escaneado, (b) Segundo Retângulo Livre Escaneado e (c) Terceiro Retângulo Livre Escaneado .....	100
<b>Figura 31</b> - Processo de Busca de um Algoritmo de Busca em Vizinhança .....	103
<b>Figura 32</b> - Algoritmo VND .....	105
<b>Figura 33</b> - Algoritmo RVNS .....	108
<b>Figura 34</b> - Algoritmo BVNS .....	109
<b>Figura 35</b> - Algoritmo GVNS.....	110
<b>Figura 36</b> - Placa Após o Corte dos Blocos na Sequência de Peças Dada pelo Caso 3 do Sistema II. (a)Após o Corte do Bloco da Peça Tipo 5 com Dois Exemplares. (b)Após o Corte do Bloco da Peça Tipo 1 com Dois Exemplares. ....	133
<b>Figura 37</b> - Comparação de Performance dos Métodos Exatos Beasley (1985) e Sistema I – Casos 1 ao 16.....	134
<b>Figura 38</b> - Comparação de Performance dos Métodos Exatos Beasley (1985) e Sistema I – Casos 17 ao 21.....	134
<b>Figura 39</b> - Comparação de Performance Entre os Algoritmos GRASP, Sistema II e Sistema III – Casos 1 ao 16.....	135
<b>Figura 40</b> - Comparação de Performance Entre os Algoritmos GRASP, Sistema II e Sistema III – Casos 17 ao 21.....	135
<b>Figura 41</b> - Planejamento de Corte do Caso de Teste 8. (a) Corte do Bloco da Peça Tipo 5 com Dois Exemplares. (b) Corte do Bloco da Peça Tipo 2 com Um Exemplar. (c) Corte do Bloco da Peça Tipo 4 com Um Exemplar. (d) Corte do Bloco da Peça Tipo 3 com Um Exemplar. (e) Corte do Bloco da Peça Tipo 1 com Três Exemplares. ....	138

## LISTA DE TABELAS

<b>Tabela 1</b> - Representação do Padrão de Corte Proposto como Solução Codificada por Lai e Chan (1997) .....	43
<b>Tabela 2</b> - Representação da Proposta de Solução Codificada por Alvarez-Valdes, Parreño e Tamarit (2007) .....	55
<b>Tabela 3</b> - Recombinação de Soluções .....	71
<b>Tabela 4</b> - Analogia Entre Sistemas Naturais e os Algoritmos Genéticos.....	80
<b>Tabela 5</b> - Valores Recomendados dos Parâmetros do Algoritmo BRKGA .....	92
<b>Tabela 6</b> - Resultados Computacionais dos 21 Casos de Teste Usando o Modelo Matemático de Hadjiconstantinou e Christofides (1995), Programado no AMPL, Considerando a Orientação Fixa das Peças. ....	120
<b>Tabela 7</b> - Análise dos Resultados Encontrados Durante os Vinte Testes Realizados – Sistema II. ....	121
<b>Tabela 8</b> - Resultados Computacionais dos 21 Casos de Teste Usando o Algoritmo Genético de Chaves Aleatórias Viciadas Especializado Considerando a Orientação Fixa das Peças. ....	122
<b>Tabela 9</b> - Resultados Computacionais dos 21 Casos de Teste Usando o Algoritmo RVNS Especializado Considerando a Orientação Fixa das Peças. ....	124
<b>Tabela 10</b> - Análise dos Resultados Encontrados Durante os Vinte Testes Realizados – Sistema IV. ....	127
<b>Tabela 11</b> - Resultados Computacionais dos 21 Casos de Teste com o Algoritmo Genético de Chaves Aleatórias Viciadas Especializado Considerando a Orientação Livre das Peças. ....	127
<b>Tabela 12</b> - Comparação de Resultados Computacionais Considerando Orientação Fixa...	130
<b>Tabela 13</b> - Comparação de Resultados Computacionais Considerando Orientação Fixa e Livre.....	136

## LISTA DE ABREVIATURAS E SIGLAS

AHC	Algoritmo Heurístico Construtivo
AMPL	A Modeling Language for Mathematical Programming
BBL	Back-Bottom-Left
BLB	Back-Left-Bottom
BPS	Box Packing Sequence
BRKGA	Biased Random Key Genetic Algorithm
C&P	Corte e Empacotamento
CPLEX	Software de Otimização
EMS	Empty Maximal-Spaces
FORTRAN	Linguagem de Programação
GRASP	Meta-heurística “Greedy Randomized Adaptive Search Procedure”
IA	Inteligência Artificial
LAYOUT	Desenho, disposição
NP	Não Polinomial
PSO	Meta-heurística “Particle Swarm Optimization”
RKGA	Random Key Genetic Algorithm
RL	Retângulos Livres
RVNS	Reduced Variable Neighbourhood Search
TS	Meta-heurística “Tabu Search”
VBO	Vector of Box Orientations
VNS	Meta-heurística “Variable Neighbourhood Search”
VPH	Vector of Placement Heuristics

## SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO</b> .....	14
<b>2</b>	<b>ANÁLISE DO PROBLEMA DE CORTE BIDIMENSIONAL NÃO GUILHOTINADO</b> .....	17
2.1	INTRODUÇÃO .....	17
2.2	MODELAGEM MATEMÁTICA .....	20
<b>2.2.1</b>	<b>Modelagem Matemática de Hadjiconstantinou-Christofides</b> .....	25
2.3	TIPOLOGIA DOS PROBLEMAS DE CORTE E EMPACOTAMENTO .....	28
2.4	TÉCNICAS DE SOLUÇÃO.....	31
2.5	ANÁLISE DO ESTADO DA ARTE DO PROBLEMA DE CORTE BIDIMENSIONAL.....	33
<b>3</b>	<b>REPRESENTAÇÃO DE PROPOSTA DE SOLUÇÃO PARA O PROBLEMA DE CORTE BIDIMENSIONAL NÃO GUILHOTINADO</b> .....	39
3.1	VISÃO GERAL .....	39
3.2	REPRESENTAÇÃO PROPOSTA POR LAI E CHAN .....	41
<b>3.2.1</b>	<b>Representação da Proposta de Solução</b> .....	41
<b>3.2.2</b>	<b>Algoritmo Heurístico Construtivo</b> .....	43
<b>3.2.2.1</b>	<b><i>Geração dos Retângulos Livres</i></b> .....	45
3.2.2.1.1	Processo de Diferença.....	46
3.2.2.1.2	Processo de Eliminação .....	48
<b>3.2.2.2</b>	<b><i>Seleção do Retângulo Livre</i></b> .....	48
<b>3.2.2.3</b>	<b><i>Geração do Padrão de Corte</i></b> .....	50
<b>3.2.3</b>	<b>Definição de Estrutura de Vizinhança</b> .....	51
3.3	REPRESENTAÇÃO PROPOSTA POR ALVAREZ-VALDES, PARREÑO E TAMARIT .....	54
<b>3.3.1</b>	<b>Representação da Proposta de Solução</b> .....	54
<b>3.3.2</b>	<b>Algoritmo Heurístico Construtivo</b> .....	56
<b>3.3.3</b>	<b>Definição de Estrutura de Vizinhança</b> .....	59
<b>3.3.3.1</b>	<b><i>Retirada de Bloco</i></b> .....	60
<b>3.3.3.2</b>	<b><i>Inserção de Bloco</i></b> .....	62
3.4	REPRESENTAÇÃO PROPOSTA POR GONÇALVES E RESENDE.....	65
<b>3.4.1</b>	<b>Representação da Proposta de Solução</b> .....	65
<b>3.4.2</b>	<b>Algoritmo Determinístico</b> .....	66
<b>3.4.3</b>	<b>Definição de Recombinação de Soluções</b> .....	70
3.5	REPRESENTAÇÃO PROPOSTA PELA AUTORA.....	71
<b>3.5.1</b>	<b>Representação da Proposta de Solução</b> .....	71
<b>4</b>	<b>ALGORITMO GENÉTICO ESPECIALIZADO PARA O PROBLEMA DE CORTE BIDIMENSIONAL NÃO GUILHOTINADO</b> .....	77
4.1	INTRODUÇÃO .....	77
4.2	FUNCIONAMENTO DO ALGORITMO GENÉTICO TRADICIONAL.....	79

4.2.1	<b>Sistema Natural x Algoritmo Genético</b> .....	79
4.2.2	<b>Funcionamento do Algoritmo Genético</b> .....	82
4.2.2.1	<i>Codificação</i> .....	82
4.2.2.2	<i>Função Objetivo</i> .....	82
4.2.2.3	<i>Métodos de Seleção</i> .....	83
4.2.2.4	<i>Métodos de Recombinação</i> .....	84
4.2.2.5	<i>Operador de Mutação</i> .....	86
4.2.2.6	<i>Parâmetros dos Algoritmos Genéticos</i> .....	86
4.2.2.7	<i>Critérios de Parada</i> .....	87
4.3	<b>ALGORITMO GENÉTICO DE CHAVES ALEATÓRIAS VICIADAS (GONÇALVES E RESENDE)</b> .....	88
4.3.1	<b>Algoritmo Genético de Chaves Aleatórias</b> .....	88
4.3.2	<b>Algoritmo Genético de Chaves Aleatórias Viciadas</b> .....	89
4.4	<b>ALGORITMO GENÉTICO DE CHAVES ALEATÓRIAS VICIADAS ESPECIALIZADO</b> .....	94
4.4.1	<b>Funcionamento do Algoritmo Genético de Chaves Aleatórias Viciadas Especializado</b> .....	94
4.4.2	<b>Formação dos Blocos</b> .....	98
4.4.3	<b>Escaneamento de Retângulos Livres</b> .....	98
5	<b>ALGORITMO RVNS ESPECIALIZADO PARA O PROBLEMA DE CORTE BIDIMENSIONAL NÃO GUILHOTINADO</b> .....	101
5.1	<b>INTRODUÇÃO À META-HEURÍSTICA DE BUSCA EM VIZINHANÇA VARIÁVEL</b> .....	101
5.2	<b>FUNDAMENTOS DE VNS</b> .....	103
5.3	<b>VNS DE DESCIDA (VND)</b> .....	105
5.4	<b>VNS REDUZIDA (RVNS)</b> .....	106
5.5	<b>VNS BÁSICA (BVNS)</b> .....	108
5.6	<b>VNS GERAL (GVNS)</b> .....	110
5.7	<b>ALGORITMO RVNS ESPECIALIZADO</b> .....	112
5.7.1	<b>Funcionamento do Algoritmo RVNS Especializado</b> .....	112
6	<b>RESULTADOS E DISCUSSÕES</b> .....	117
6.1	<b>TESTES</b> .....	117
6.2	<b>RESULTADOS DOS TESTES COM O SISTEMA I</b> .....	119
6.3	<b>RESULTADOS DOS TESTES COM O SISTEMA II</b> .....	121
6.4	<b>RESULTADOS DOS TESTES COM O SISTEMA III</b> .....	124
6.5	<b>RESULTADOS DOS TESTES COM O SISTEMA IV</b> .....	127
6.6	<b>ANÁLISE E DISCUSSÕES SOBRE OS RESULTADOS</b> .....	130
7	<b>CONCLUSÕES</b> .....	141
	<b>REFERÊNCIAS</b> .....	143
	<b>APÊNDICE A – Dados da pesquisa obtidos na OR-Library (2017, online)</b> .....	147

## 1 INTRODUÇÃO

No estágio de globalização em que se encontram os mercados atualmente, produtos com preços competitivos têm sido cada vez mais almejados pelas indústrias. Para tanto, tem-se investido na otimização de processos industriais a fim de se atingir a competitividade necessária. Uma pequena redução dos custos nas operações industriais implicam uma economia de recursos financeiros, conseqüentemente favorecem a competitividade.

Vários tipos de problemas surgem quando se trata de otimizar processos produtivos, dentre eles destaca-se o problema de corte bidimensional não guilhotinado, o qual consiste em cortar peças retangulares de uma placa retangular maior, podendo ser aço, tecido, papel, vidro, madeira ou, até mesmo, colocar anúncios nas páginas de jornais e revistas, de forma a maximizar o valor total das peças cortadas, ou de minimizar o desperdício de material. Somente duas dimensões são consideradas pelo problema em estudo, e a técnica de corte a laser é utilizada no referido problema em vez da tradicional guilhotina. O problema possui várias aplicações industriais sempre que pequenas peças têm que ser cortadas de um material retangular maior, segundo Alvarez-Valdes, Parreño e Tamarit (2007).

Muitas vezes, o que se vê na indústria são peças com alto valor agregado, as quais devem ser cortadas da placa (matéria-prima), porém essas peças isoladas não significam o produto acabado, pronto para ser entregue. Logo, para que o produto seja finalizado pela indústria, tal peça, com alto valor, deve interagir com outras cujo valor é menor, contudo não menos importante para a entrega do produto final. Diante dessa situação, a indústria de corte de matéria-prima vê-se obrigada a cortar as peças de menor valor. Não se resolve o problema cortando somente as peças mais lucrativas, visto que elas interagem e compõem um produto que, para ser finalizado, necessita de outras peças menos lucrativas, portanto o problema é complexo, pois envolve várias questões, a saber: quantas cópias cortar de cada peça, como alocá-las na placa a fim de que se tenha uma maior quantidade de peças cortadas e, ainda, como reduzir o desperdício da matéria-prima.

Devido à sua importância econômica, o problema de corte bidimensional não guilhotinado tornou-se relevante para a Pesquisa Operacional, visto que se trata de um problema que apresenta facilidade para ser representado por meio de modelos matemáticos, mas que, devido à sua complexidade, torna-se difícil de ser solucionado. A importância do problema motivou a realização desta pesquisa, por conseguinte o estudo de técnicas de solução e o desenvolvimento de algoritmos que o resolvessem.



Neste trabalho teve-se como objetivo geral estudar as meta-heurísticas aplicadas na resolução do problema de corte bidimensional não guilhotinado. Em particular, foram estudados os artigos de Alvarez-Valdes, Parreño e Tamarit (2007) onde foi utilizada a meta-heurística *Tabu Search* para a resolução do problema, outro artigo estudado foi de Lai e Chan (1997), onde a meta-heurística *Simulated Annealing* foi escolhida para a resolução do problema, além do artigo de Gonçalves e Resende (2013) no qual foi aplicado a meta-heurística Algoritmo Genético para resolver o problema.

Como objetivo específico do trabalho, propôs-se aplicar duas meta-heurísticas na resolução do problema: a primeira delas foi o Algoritmo Genético de Chaves Aleatórias Viciadas, proposto por Gonçalves e Resende (2013), acrescentando melhorias em sua implementação; a segunda, a RVNS (*Reduced Variable Neighbourhood Search*), utilizando um algoritmo RVNS, especializado na resolução do problema de corte bidimensional não guilhotinado. Propõe-se também, entre os objetivos específicos, a realização de uma análise dos diferentes formatos de representação da proposta de solução do problema em estudo utilizados por autores da área e, ainda, uma nova forma de representação da proposta de solução do problema, outro objetivo específico foi a realização de uma análise comparativa entre os resultados encontrados, na literatura, sobre algoritmos meta-heurísticos aplicados ao problema em estudo e os resultados encontrados pelas duas meta-heurísticas implementadas aqui. Uma terceira comparação foi realizada, utilizando-se os resultados encontrados pelo modelo matemático de Hadjiconstantinou e Christofides (1995) para a resolução do problema de corte bidimensional não guilhotinado.

O texto desta tese está estruturado em 7 capítulos. No capítulo 1, apresentam-se a introdução do problema de corte bidimensional não guilhotinado e os objetivos almejados nesta pesquisa. No capítulo 2, são apresentados, de maneira formal, o problema de corte bidimensional e suas variantes, bem como a modelagem matemática do problema e as técnicas de solução que estão sendo aplicadas na resolução do problema. No capítulo 3, encontram-se a descrição das representações de proposta de solução encontradas na literatura e a proposta desta pesquisa para fornecer uma proposta de solução diferenciada das demais. No capítulo 4, é descrito o algoritmo genético de chaves aleatórias viciadas especializado para otimizar o problema de corte bidimensional. Nesse capítulo, também é abordado o funcionamento do algoritmo genético tradicional, apresentada uma descrição do algoritmo genético de chaves aleatórias viciadas e, por fim, são elencadas as melhorias propostas para o algoritmo genético de chaves aleatórias viciadas, especializado para o problema de corte bidimensional não guilhotinado. No capítulo 5, é descrito o algoritmo RVNS, especializado na otimização do

problema de corte bidimensional. Nesse capítulo, também são abordados os fundamentos de Busca em Vizinhança Variável, o esquema básico de VNS básica, VNS de descida, VNS reduzida e VNS geral. No capítulo 6, são descritos os resultados encontrados pelos algoritmos meta-heurísticos implementados neste trabalho, bem como os resultados encontrados pelo modelo matemático de Hadjiconstantinou e Christofides (1995) ao aplicar os testes das diferentes instâncias conhecidas na literatura. Neste capítulo é realizada uma comparação entre os resultados encontrados pelos algoritmos e o modelo matemático implementado neste trabalho e também é realizada uma análise e discussão sobre os resultados. No capítulo 7, são descritas as conclusões e as considerações finais do trabalho. Posteriormente, ao final do documento, encontram-se as referências utilizadas e os apêndices com os dados de cada sistema testado.

## 2 ANÁLISE DO PROBLEMA DE CORTE BIDIMENSIONAL NÃO GUILHOTINADO

Neste capítulo, encontram-se a definição formal do problema de corte bidimensional não guilhotinado, da tipologia dos problemas de corte e de empacotamento, bem como as técnicas de solução e a análise do estado da arte do problema em estudo.

### 2.1 INTRODUÇÃO

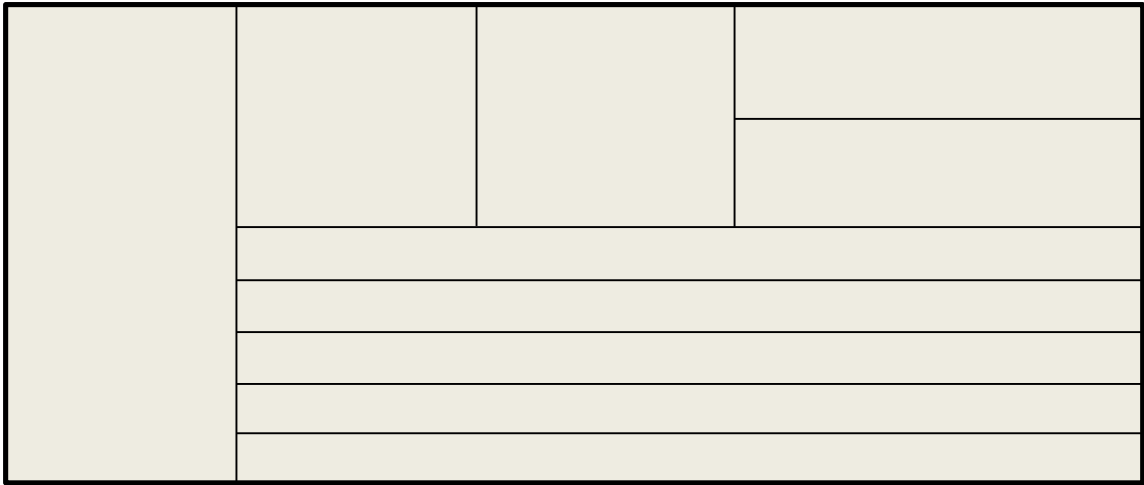
Os problemas de corte, em geral, são aqueles que requerem que peças menores sejam cortadas de placas maiores, de forma a maximizar o valor total das peças cortadas, ou de minimizar o desperdício de material. Segundo Alvarez-Valdes, Parreño e Tamarit (2007), essas operações de corte são frequentes em diversas indústrias, tais como as de madeira, tecido, vidro, aço, espuma, entre outras. O problema de corte foi introduzido, na literatura, por Gilmore e Gomory (1961). O problema consiste, basicamente, em determinar a melhor maneira de cortar placas maiores a fim de se obter peças menores, respeitando-se determinadas restrições e minimizando-se as perdas ou maximizando-se a utilização da placa.

Entre os problemas de corte, existem os que são guilhotinados e os não guilhotinados. Em várias indústrias, o equipamento de corte opera de tal forma que qualquer corte feito sobre um retângulo deve ser realizado em uma linha reta, de uma borda da placa a outra. Esse tipo de corte é referido como corte de guilhotina (LAI; CHAN, 1997). No entanto, em algumas aplicações, o corte de guilhotina não é o mais indicado, o que viabiliza a utilização de uma outra técnica de corte chamada corte a laser, permitindo o corte não guilhotinado.

Beasley (1985) também explica esse tipo de corte em seu trabalho. Segundo o autor, para ser considerado um problema de corte guilhotinado, a guilhotina deve cortar uma placa de uma extremidade à outra, em paralelo com as duas arestas restantes. Portanto, é chamado guilhotinado, porque as peças devem ser de um tipo de guilhotina, devido às limitações impostas pelo equipamento de corte.

Uma guilhotina ocorre quando um corte aplicado a um retângulo produz dois retângulos (MORABITO; PUREZA, 2007). Esse tipo de problema traz restrições em relação a quais peças poderão ser cortadas depois do primeiro e do segundo corte, e assim sucessivamente, limitando o conjunto de peças que poderão ser cortadas, se essas peças forem de tamanhos diferentes. Ao mesmo tempo, se o problema envolve peças do mesmo tamanho, pode ser mais eficiente o corte guilhotinado. Na Figura 1 ilustra-se o problema de corte guilhotinado.

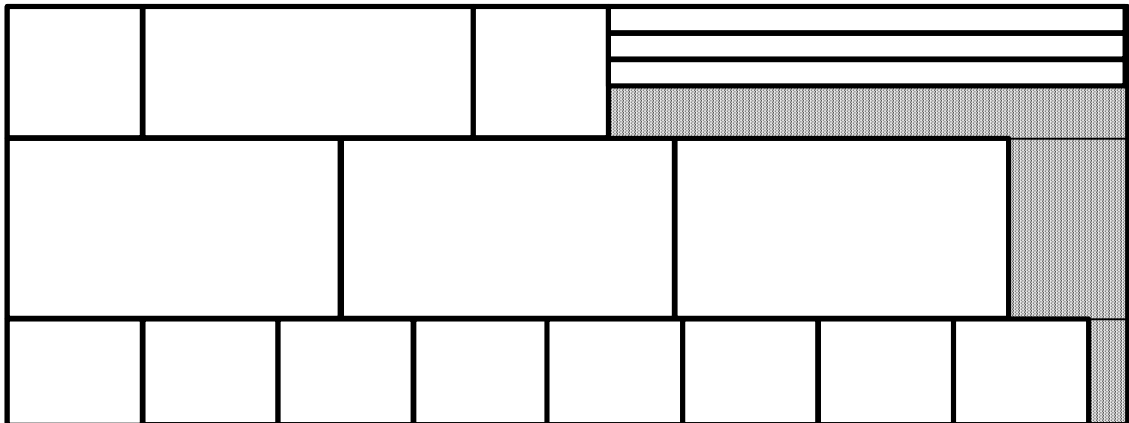
**Figura 1** - Exemplo de Corte Guilhotinado



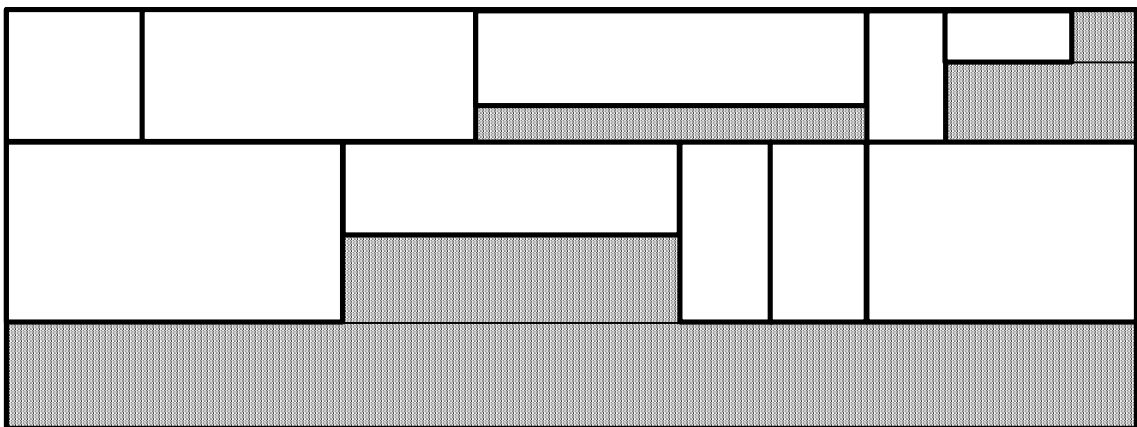
Fonte: Elaborado pela autora.

Na Figura 2, são ilustrados os diferentes estilos de padrões de cortes guilhotinados.

**Figura 2** – Exemplo de Diferentes Estilos de Padrões de Cortes



*(a) Um padrão de Corte Guilhotinado.*

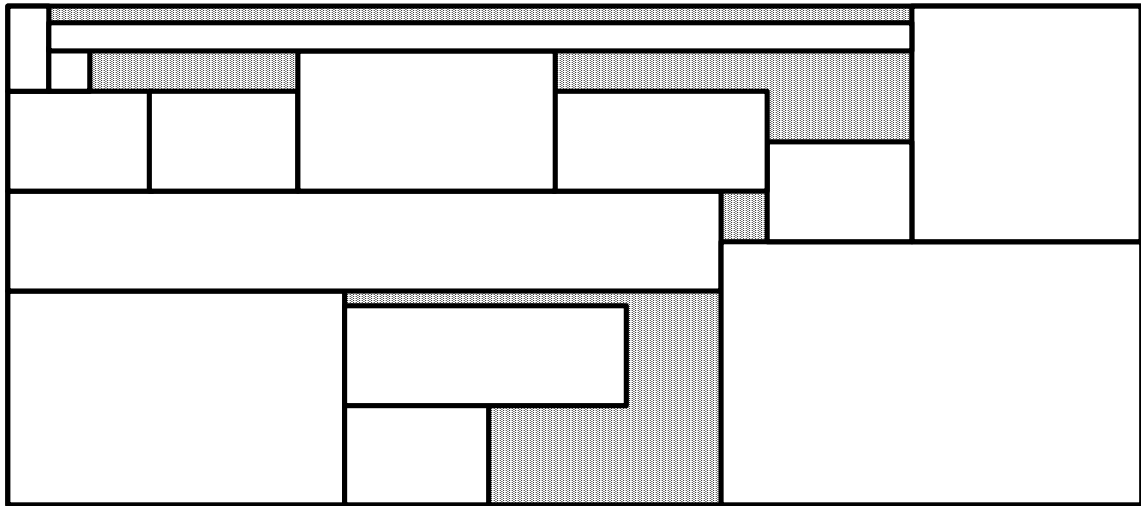


*(b) Outro Padrão de Corte Guilhotinado.*

Fonte: Elaborado pela autora.

O problema de corte não guilhotinado não segue o padrão de corte de uma guilhotina, podendo ter recortes diferentes para peças com dimensões diferentes. Na Figura 3, ilustra-se o caso do problema de corte não guilhotinado.

**Figura 3 - Exemplo de Corte Não Guilhotinado**



Fonte: Elaborado pela autora.

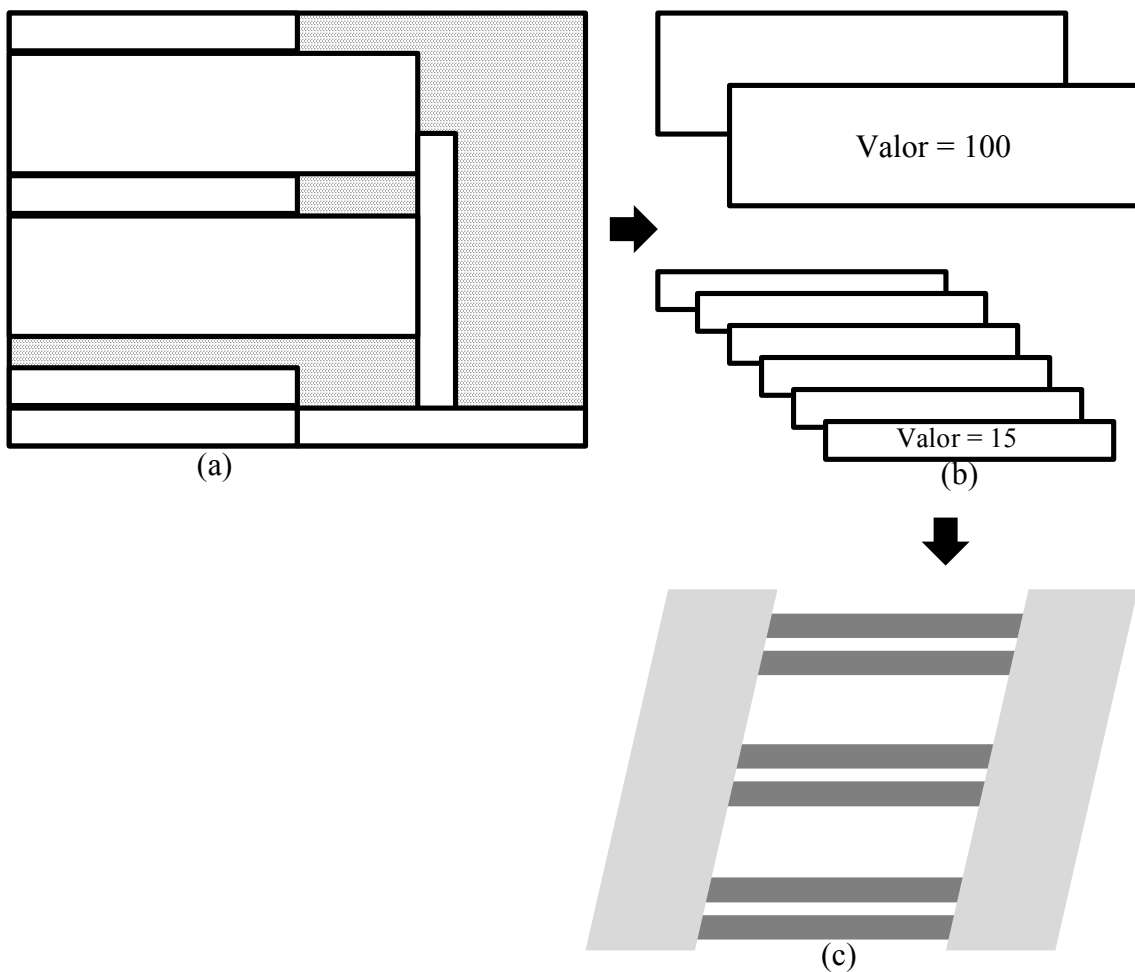
Neste trabalho é avaliado o problema de corte bidimensional não guilhotinado, em que se tem uma variedade finita de peças retangulares de tamanhos diferentes. Cada uma dessas peças possui seu valor correspondente e uma quantidade finita de cópias de peças disponíveis para serem cortadas em uma placa maior, de tamanho retangular finito. O problema aparece, em muitos processos industriais, quando um grande retângulo tem que ser cortado em partes menores para satisfazer os requisitos do cliente, porém somente duas dimensões são consideradas, além de se utilizar a técnica de corte a laser.

Na prática, se a empresa que precisa aplicar o corte em uma placa possui diversas peças de formatos e de valores diferentes, as peças com valores maiores tornam-se mais interessantes, todavia, ao mesmo tempo, deve-se compreender que, muitas vezes, essas peças fazem parte de um contexto de interação com outras de valores menores, tendo, assim, o fabricante que tomar o cuidado de produzir todas elas. Se não fosse assim, as indústrias somente cortariam as peças que gerariam maior lucro e, por conta desse impasse, existem os limitantes inferiores e superiores aplicados na quantidade de cada peça a ser cortada. Os limitantes inferiores traduzem a quantidade mínima de cópias que cada peça pode ter no corte da placa, e os limitantes superiores trazem a quantidade máxima de cópias que cada peça pode apresentar no corte da

placa, ou seja, por mais que seja interessante para a indústria cortar as peças mais valiosas, deve-se respeitar as restrições dos dois limitantes.

Na Figura 4, o problema é ilustrado permitindo um melhor entendimento de sua prática na indústria. Pode-se verificar, na Figura 4(a), que as peças menores são cortadas em uma placa maior, cada uma delas tem um valor para a indústria (Figura 4(b)) e dá origem a um produto final (Figura 4(c)).

**Figura 4** - Exemplo do Corte de Placa Retangular Maior em Peças Retangulares Menores. (a) Padrão de Corte de Peças Menores em uma Placa Maior. (b) Peças Resultantes do Corte da Placa. (c) Produto Finalizado pela Indústria



Fonte: Elaborado pela autora.

## 2.2 MODELAGEM MATEMÁTICA

Em Beasley (1985) encontra-se uma definição formal para o problema de corte bidimensional não guilhotinado. Segundo esse autor, quando uma placa retangular necessita ser cortada em várias partes retangulares de menor dimensão, de tamanho determinado, e cada uma

com seu valor correspondente, o objetivo, nesse caso, é maximizar o valor do produto final, sujeito à restrição de que o número de cada peça cortada deve estar dentro dos limitantes preestabelecidos.

Uma outra definição do problema de corte bidimensional encontra-se em Alvarez-Valdes, Parreño e Tamarit (2007). Para esses autores, o problema consiste em cortar um determinado conjunto finito de peças retangulares de uma placa de dimensões fixas, com o máximo de lucro, pois cada peça tem um valor associado a ela. A placa  $R$  possui comprimento  $C$  e altura  $A$ . Cada tipo de peça  $i$  tem as dimensões:  $c_i$ , para comprimento da peça, e  $a_i$ , para a altura da peça, e um valor  $v_i$ . O problema é cortar a placa  $R$  em  $x_i$  cópias de cada peça  $i$ , onde  $i$  vai de 1 a  $m$ , tal que a quantidade de cópias  $x_i$  de cada peça  $i$  não seja menor que seu limitante inferior,  $P_i$ , onde este limitante pode chegar a zero para algumas peças  $i$ , e também não ultrapasse seu limitante superior,  $Q_i$ , e o valor total de peças cortadas  $\sum_i v_i x_i$ , ou seja, a somatória dos valores das peças cortada em  $R$  é o valor que se busca maximizar nesse problema.

Uma sequência de cortes em  $R$  é chamada de padrão de corte. Em alguns casos, tem-se a imposição de duas restrições adicionais para os padrões de corte:

- As peças têm orientação fixa, ou seja, peças de dimensões  $(a, b)$  e  $(b, a)$  são consideradas diferentes.
- Cada peça deve ser cortada com suas bordas paralelas às bordas da placa (cortes ortogonais).

Segundo Alvarez-Valdes, Parreño e Tamarit (2007), de acordo com os valores dos limitantes inferiores,  $P_i$ , e limitantes superiores,  $Q_i$ , das peças  $i$  a serem cortadas na placa  $R$ , podem-se distinguir três tipos de problemas: irrestrito, restrito e duplamente restrito.

O primeiro problema é chamado de irrestrito ou sem restrições, pois não existem restrições inferiores e superiores a serem cumpridas. A única restrição está relacionada a respeitar os limites da área da placa  $R$ . Essa primeira situação ocorre quando para todas as peças  $i$  a serem cortadas em  $R$  não se determina um número mínimo, nem um número máximo de cópias, ou seja, pode existir uma peça  $i$  que não tenha seu exemplar cortado na placa  $R$ , assim como pode existir uma peça  $i$  onde todos os seus exemplares foram cortados na placa  $R$ , podendo ocupar totalmente a área da placa.

No problema restrito, existe uma única restrição com relação ao limitante superior. Nesse problema, todas as peças  $i$ , a serem cortadas em  $R$ , não apresentam um número mínimo de cópias a serem cortadas, todavia existe a restrição com relação ao número máximo de cópias para peças  $i$  quaisquer.

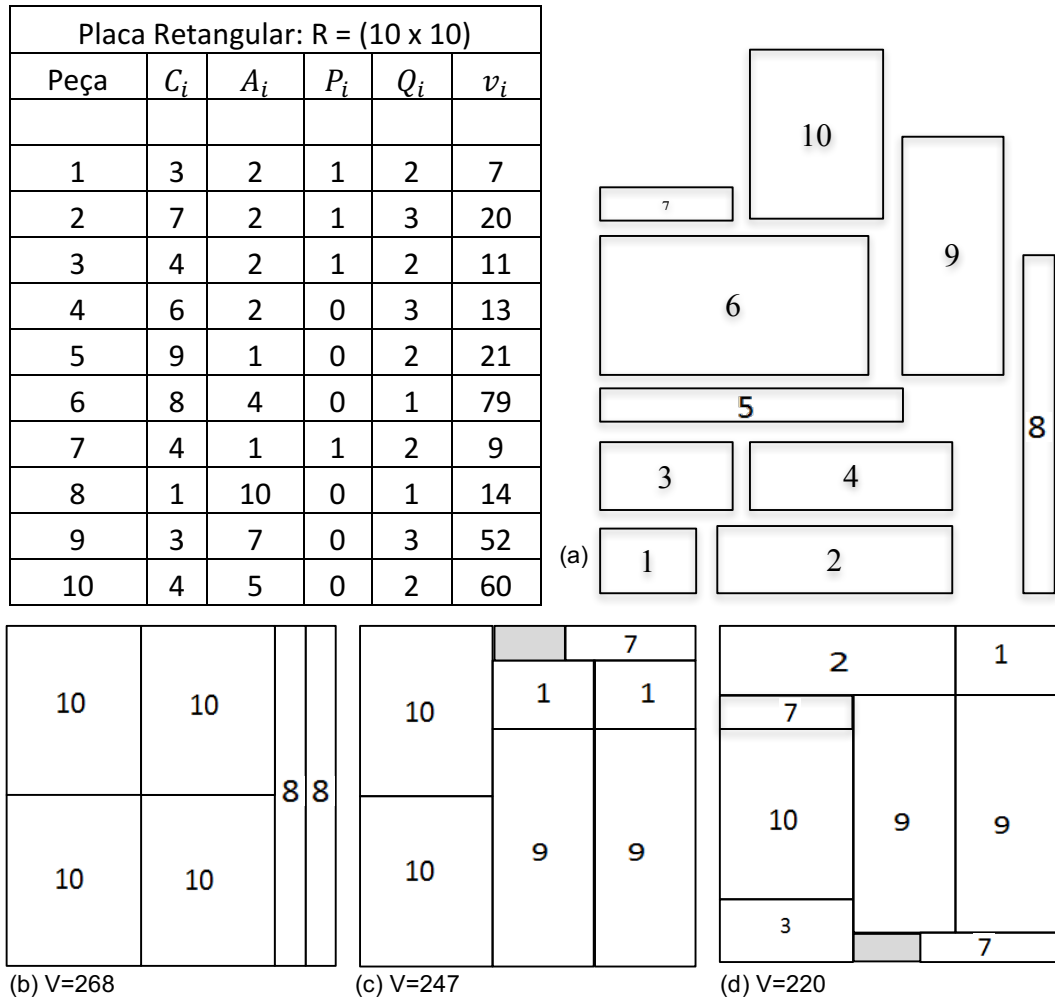
O limitante superior indica que de uma placa  $R$  podem ser cortados até  $Q_i$  cópias da peça  $i$ . Se a somatória das áreas das  $Q_i$  cópias da peça  $i$  for menor que a área da placa  $R$ , e todos os exemplares da peça  $i$  já foram utilizados no corte da placa, mesmo existindo a possibilidade de cortar mais cópias desta peça  $i$ , não poderá ser feito, pois foi atingido o limitante superior e esta restrição deve ser respeitada. Do mesmo modo que se a somatória das áreas das  $Q_i$  peças  $i$  for maior que a área da placa  $R$ , não serão utilizadas todas as cópias desta peça, dessa maneira a restrição continua sendo respeitada, porém sem utilizar todas as cópias especificadas pelo limitante superior.

O último tipo de problema é o duplamente restrito, quando existem limitantes inferiores e superiores que devem ser respeitados por peças  $i$  quaisquer a serem cortadas na placa  $R$ . Na prática, esse problema ocorre quando são cortadas, na placa  $R$ , obrigatoriamente,  $P_i$  cópias de uma peça  $i$  qualquer, onde  $P_i$  representa o limitante inferior desta peça, sendo este número maior que zero. A somatória das áreas das  $P_i$  cópias da peça  $i$  não ultrapassará o total da área da placa  $R$ , respeitando, assim, a primeira restrição com relação ao limitante inferior. Em conformidade com a dupla restrição do problema, a mesma peça  $i$  apresenta um número máximo de cópias a serem cortadas em  $R$ , desde que a somatória das áreas das  $Q_i$  cópias da peça  $i$  não ultrapasse a área da placa  $R$ , é possível cortar uma quantidade de exemplares da peça  $i$  entre o intervalo do valor do seu limitante inferior até o valor do seu limitante superior, respeitando-se, dessa forma, a segunda restrição e, portanto, seguindo as duas restrições impostas por esse último problema.

Na Figura 5, ilustra-se um exemplo de uma placa ( $10 \times 10$ ) e de  $m = 10$  peças a serem cortadas. A primeira solução (Figura 5(b)) é ótima para o problema irrestrito, enquanto que a segunda solução (Figura 5(c)) corresponde aos problemas restritos e a terceira solução (Figura 5(d)), para o problema duplamente restrito.



**Figura 5** - Exemplo de uma Instância Usada para Testar o Problema de Corte Bidimensional Não Guilhotinado. (a) Instância de Beasley (1985). (b) Irrestrito com Otimização: 268. (c) Restrito com Otimização: 247. (d) Duplamente Restrito com Otimização: 220



Fonte: Adaptado de Beasley (1985).

Um limitante superior do valor da função objetivo do problema de corte bidimensional não guilhotinado com restrição pode ser obtido resolvendo o seguinte problema da mochila limitada, onde a variável  $x_i$  representa o número de peças do tipo  $i$  a serem cortadas acima do valor de seu limitante inferior  $P_i$ :

$$\text{Max} \sum_{i=1}^m v_i x_i + \sum_{i=1}^m v_i P_i \quad (1)$$

s. a.:

$$\sum_{i=1}^m (c_i a_i) x_i \leq CA - \sum_{i=1}^m P_i (c_i a_i), \quad (2)$$

$$x_i \leq Q_i - P_i, i = 1, \dots, m, \quad (3)$$

$$x_i \geq 0, \text{inteiro}, i = 1, \dots, m. \quad (4)$$

Analisando-se as restrições (3) e (4), é possível descrever que  $x_i$  é o número de cópias que cada tipo de peça  $i$  pode cortar na placa  $R$ , acima do valor do seu limitante inferior e não ultrapassando o valor do seu limitante superior, portanto  $x_i$  é um número de cópias excedentes que deve estar entre os dois limitantes de cada peça  $i$  cortada em  $R$ . O número  $x_i$  deve ser inteiro e  $i$  representa os tipos de peças que serão utilizadas no problema, seu valor vai de 1 até  $m$ .

Avaliando-se a função objetivo (1) e a restrição (2), é possível entender que o objetivo do problema é maximizar a somatória dos valores dos exemplares das peças  $i$  cortadas em  $R$ , onde é considerado como obrigatório o corte de exemplares das peças  $i$  que apresentam um valor diferente de zero em seu limitante inferior  $P_i$ , portanto tais peças terão seus valores somados na função objetivo (1). Além dos valores dos exemplares de peças  $i$  que são consideradas obrigatórias no corte em  $R$ , adiciona-se a soma dos valores dos exemplares excedentes das peças  $i$  cujas áreas não ultrapassem a área livre da placa  $R$ , resultante da exclusão da área utilizada pelas cópias de peças  $i$  consideradas obrigatórias.

Alguns autores têm considerado o problema irrestrito, são eles: Tsai, Malstrom e Meeks (1988), Morabito e Arenales (1995), Healy, Creavin e Kuusik (1999). No entanto, segundo Alvarez-Valdes, Parreño e Tamarit (2005), os problemas com restrições são mais interessantes para as aplicações reais e, cada vez mais, pesquisas têm se dedicado a essa classe de problemas.

Ainda segundo Alvarez-Valdes, Parreño e Tamarit (2007), várias heurísticas têm sido propostas. Wu et al. (2002 apud ALVAREZ-VALDES, PARREÑO E TAMARIT, 2007) desenvolveram um algoritmo heurístico construtivo para o caso especial em que o limitante inferior é igual ao limitante superior para toda peça  $i$  a ser cortada em  $R$ . Em cada passo, uma peça é cortada em um canto do padrão de corte corrente, e a peça a ser cortada é escolhida de acordo com uma função *fitness*, função de avaliação que estima a qualidade completa da solução que poderia ser obtida a partir da peça que está sendo considerada. Dessa forma,

verifica-se que o problema de corte bidimensional não guilhotinado é objeto de estudo de diferentes autores, sendo que cada autor propõe uma solução que será discutida nas próximas subseções.

### 2.2.1 Modelagem Matemática de Hadjiconstantinou-Christofides

Hadjiconstantinou e Christofides (1995), no artigo titulado “Um algoritmo exato para o problema geral, ortogonal e bidimensional da mochila”, publicado em 1995, apresentaram uma modelagem matemática do problema da mochila que foi resolvido por um procedimento exato de busca em árvore.

Segundo os autores, o problema de cortar uma placa retangular plana em peças retangulares menores (cada peça tendo um dado tamanho e valor), de forma a maximizar o valor total das peças cortadas, é conhecido como o “problema da mochila” e, na literatura técnica, também conhecido como o “problema de corte bidimensional”.

Hadjiconstantinou e Christofides (1995) definiram o problema de corte bidimensional em que um número de pequenas peças retangulares, cada uma de um determinado tamanho e valor, devem ser cortadas de uma placa retangular maior. O objetivo é maximizar o valor das peças cortadas ou minimizar o desperdício, se for tomado o valor de uma peça como sendo proporcional à sua área. Os referidos autores consideraram o caso restrito do problema, onde há um limitante superior determinando o número máximo de vezes que uma peça pode ser usada em um padrão de corte.

Peças retangulares podem ser cortadas em padrões de corte ortogonais ou não ortogonais, segundo Hadjiconstantinou e Christofides (1995). Em padrões ortogonais, as bordas das peças são dispostas paralelamente às bordas da placa, ao passo que, nos padrões não-ortogonais, o ângulo é opcional. Eles descreveram os casos nos quais o corte é ortogonal e guilhotinado como especiais, permitidos apenas de uma extremidade da placa à outra. Já aqueles em que os cortes são ortogonais e não guilhotinados, são considerados pelos autores como pouco explorados e, quando estudados, utilizam-se algoritmos heurísticos e meta-heurísticos como técnica de solução.

A modelagem matemática proposta por Hadjiconstantinou e Christofides (1995) é para o problema de corte bidimensional, ortogonal, não guilhotinado e restrito com relação ao limitante superior. A seguir, ela é descrita detalhadamente.

Seja  $A_0 = (\alpha_0, \beta_0)$  uma placa retangular com comprimento  $\alpha_0$  e altura  $\beta_0$  e seja  $R$  um conjunto de  $m$  peças retangulares menores  $R_1, R_2, \dots, R_m$  com dimensões  $(\alpha_1, \beta_1), (\alpha_2, \beta_2), \dots,$

$(\alpha_m, \beta_m)$  e com valores correspondentes  $v_1, v_2, \dots, v_m$  para cada peça de  $R$ . O objetivo é construir um padrão de corte não guilhotinado para  $A_0$  com o maior valor total possível, usando não mais do que  $Q_i$  exemplares de cada peça  $R_i$  para todos  $i = 1, \dots, m$ . Existe também um requisito mínimo  $P_i$  de exemplares de cada peça  $R_i$  para ser usado no padrão de corte.

Com o objetivo de distinguir entre ambos os componentes, as peças dadas no conjunto  $R$  e os retângulos livres produzidos pelos cortes em  $A_0$  em qualquer fase durante o processo de corte, os autores referem-se aos primeiros como “peças”, e aos últimos como “retângulos livres”. Hadjiconstantinou e Christofides (1995) consideraram o seguinte:

(i) Todas as dimensões  $(\alpha_i, \beta_i)$ , para  $i = 0, \dots, m$ , são assumidas de valores inteiros e, portanto, os cortes nas placas devem ser feitos em etapas inteiras ao longo dos eixos  $x$  ou  $y$ . Essa limitação não é considerada irreal pelos autores, já que, na prática, as dimensões reais podem ser ampliadas.

(ii) A orientação das peças é considerada fixa, ou seja, uma peça de comprimento  $e$  e altura  $f$  é diferente de uma peça de comprimento  $f$  e altura  $e$ .

Para formular o problema como um problema de programação inteira zero-um, tem-se

$$L_i = \{x \mid 0 \leq x \leq \alpha_0 - \alpha_i \mid x \text{ inteiro}\}$$

seja o conjunto de todos os pontos possíveis  $x$  ao longo do comprimento de  $A_0$ , de modo que qualquer peça  $R_i$  do conjunto  $R$  possa ser cortada de  $A_0$  com a sua altura paralela ao eixo  $y$  e a peça cortada tenha seu canto inferior esquerdo em qualquer  $x \in L_i$ . Da mesma forma, os autores definiram

$$W_i = \{y \mid 0 \leq y \leq \beta_0 - \beta_i \mid y \text{ inteiro}\}.$$

Os conjuntos  $L_i$  e  $W_i$  são definidos para  $i = 1, \dots, m$ . Além disso, deixa-se  $x_{ijp} = 1$  se o  $j$ -ésimo exemplar da peça  $i$  é cortado com seu canto inferior esquerdo na  $x$ -posição  $p$  onde  $j=1, \dots, Q_i$  e  $p \in L_i$  e  $x_{ijp} = 0$  em caso contrário e,  $y_{ijq} = 1$  se o  $j$ -ésimo exemplar da peça  $i$  é cortado com seu canto inferior esquerdo na  $y$ -posição  $q$  onde  $j=1, \dots, Q_i$  e  $q \in W_i$  e  $y_{ijq} = 0$  em caso contrário.

$Z_{rs} = 1$  se o ponto  $(r, s)$  em  $A_0$ , onde  $r = 0, \dots, \alpha_0 - 1$  e  $s = 0, \dots, \beta_0 - 1$ , não foi cortado por nenhuma peça do conjunto  $R$ .

$z_{rs} = 1$  se o ponto  $(r, s)$  em  $A_0$ , onde  $r = 0, \dots, \alpha_0 - 1$  e  $s = 0, \dots, \beta_0 - 1$ , não foi cortado por nenhuma peça do conjunto  $R$ .

Então, segundo Hadjiconstantinou e Christofides (1995), a formulação pode ser estabelecida da seguinte forma:

(Problema P)

$$\text{Max } Z = \sum_{i=1}^m v_i \sum_{j=1}^Q \sum_{p \in L} x_{ijp} \quad (1)$$

sujeito a

$$\sum_{s=q}^{q+\beta_1-1} \sum_{r=p}^{p+\alpha_1-1} z_{rs} \leq (2 - x_{ijp} - y_{ijq})\alpha_i\beta_i, \quad i = 1, \dots, m, \quad j = 1, \dots, Q_i, \quad p \in L_i, q \in W_i, \quad (2)$$

$$\sum_{p \in L_i} x_{ijp} \leq 1, \quad i = 1, \dots, m, \quad j = 1, \dots, Q_i, \quad (3)$$

$$\sum_{p \in L_i} x_{ijp} = \sum_{q \in W_i} y_{ijq}, \quad i = 1, \dots, m, \quad j = 1, \dots, Q_i, \quad (4)$$

$$\sum_{i=1}^m \beta_i \sum_{j=1}^{Q_i} \sum_{p=r-\alpha_1+1, p \in L_i}^r x_{ijp} + \sum_{s=0}^{\beta_0-1} z_{rs} = \beta_0, \quad r = 0, \dots, \alpha_0 - 1, \quad (5)$$

$$\sum_{i=1}^m \alpha_i \sum_{j=1}^{Q_i} \sum_{q=s-\beta_1+1, q \in W_i}^s y_{ijq} + \sum_{r=0}^{\alpha_0-1} z_{rs} = \alpha_0, \quad s = 0, \dots, \beta_0 - 1, \quad (6)$$

$$x_{ijp}, y_{ijq} \in \{0, 1\}, \quad i = 1, \dots, m, \quad j = 1, \dots, Q_i, \quad p \in L_i, \quad q \in W_i, \quad (7a)$$

$$z_{rs} \in \{0, 1\}, \quad r = 0, \dots, \alpha_0 - 1, \quad s = 0, \dots, \beta_0 - 1. \quad (7b)$$

A restrição (2) garante que qualquer ponto em  $A_0$  seja cortado em, no máximo, uma peça. As restrições (3) e (4) expressam o fato de que qualquer peça é cortada, no máximo, uma vez em  $A_0$ . A restrição (5) garante que nem todas as peças do conjunto  $R$ , cuja soma das alturas excede  $\beta_0$ , podem ser cortadas em  $A_0$  com o mesmo comprimento, tendo como origem os cantos inferiores esquerdos de cada peça. Da mesma forma, a restrição (6) assegura que se a soma dos comprimentos das peças excederem  $\alpha_0$ , então nem todas as peças poderão ser cortadas em  $A_0$  com a mesma altura, tendo como origem seus cantos inferiores esquerdos.

Este modelo é uma formulação completa do problema de corte bidimensional, ortogonal, não guilhotinado, envolvendo, aproximadamente,  $N_c$  restrições e  $N_v$  variáveis inteiras:

$$N_c = 2 \sum_{i=1}^m Q_i + \sum_{i=1}^m Q_i |L_i| |W_i| + \alpha_0 + \beta_0 \quad (8)$$

e

$$N_v = \sum_{i=1}^m Q_i (|L_i| + |W_i|) + \alpha_0 \beta_0. \quad (9)$$

Segundo Hadjiconstantinou e Christofides (1995), a dimensão dessa formulação depende do número total de peças em  $R$  e, ainda, do número máximo  $Q_i$  de exemplares de cada peça que pode ser usada para o corte.

Os autores propuseram como solução para o problema em estudo um algoritmo que limita o tamanho da busca em árvore, usando um limite superior mais restrito derivado de um relaxamento Lagrangeano de uma formulação de programação inteira de 0-1 do problema original. Um método de otimização de subgradiente foi utilizado por Hadjiconstantinou e Christofides (1995) para minimizar o limite superior resultante.

### 2.3 TIPOLOGIA DOS PROBLEMAS DE CORTE E EMPACOTAMENTO

Na literatura, encontram-se várias descrições sobre os problemas de corte e empacotamento e suas variantes; muitos autores denominam essas descrições de variantes do problema como classificação ou tipologia do problema. A tipologia fornece uma visão concisa das diferenças dos problemas dessa família, aponta as características relevantes e importantes de cada problema, auxiliando os pesquisadores na investigação orientada para a prática. Além disso, a classificação ajuda a unificar as definições e notações e, ao fazer isso, facilita a comunicação entre pesquisadores da área (WÄSCHER; HAUSSNER; SCHUMANN, 2007).

Dyckhoff (1990) classifica os problemas de corte e empacotamento (*C&P-problems*) de acordo com algumas características e tipos elementares:

- A dimensionalidade é uma delas, ou seja, se o problema é unidimensional, bidimensional ou tridimensional.
- A medição da quantidade de placas a ser utilizada é outra característica importante. Para isso, Gilmore (1979 apud DYCKHOFF, 1990) distingue dois casos: o caso discreto, que define a quantidade de placas a ser utilizada no problema, cada uma com suas dimensões definidas e finitas; e o caso contínuo, em que a placa tem como característica uma das dimensões, normalmente o comprimento ser infinito.

- O formato das peças também compõe uma das características importantes do problema em que a representação geométrica, tanto da placa quanto das peças, deve ser considerada. Além da importância deste, destacam-se o tamanho e a orientação.
- A variedade, dada por formatos e tipos de peças permitidas, é uma outra característica elencada por Dyckhoff (1990). Aqui a variedade é dada por formatos e tipos de peças permitidas.
- A disponibilidade de peças é outra característica importante na qual os limitantes inferiores e superiores devem ser considerados, assim como a sequência ou a ordem em que uma peça pode ou deve ser cortada.

Assim, Dyckhoff (1990) descreveu uma lista da tipologia do problema de corte e empacotamento:

#### 1. Dimensionalidade

- Unidimensional;
- Bidimensional;
- Tridimensional;
- N-dimensional com  $N > 3$ .

#### 2. Tipo de Atribuição

- Todas as peças e uma seleção de cópias;
- Uma seleção de peças e todas as cópias.

#### 3. Variedade de Placas

- Uma única placa;
- Placas idênticas;
- Diferentes tipos de placas.

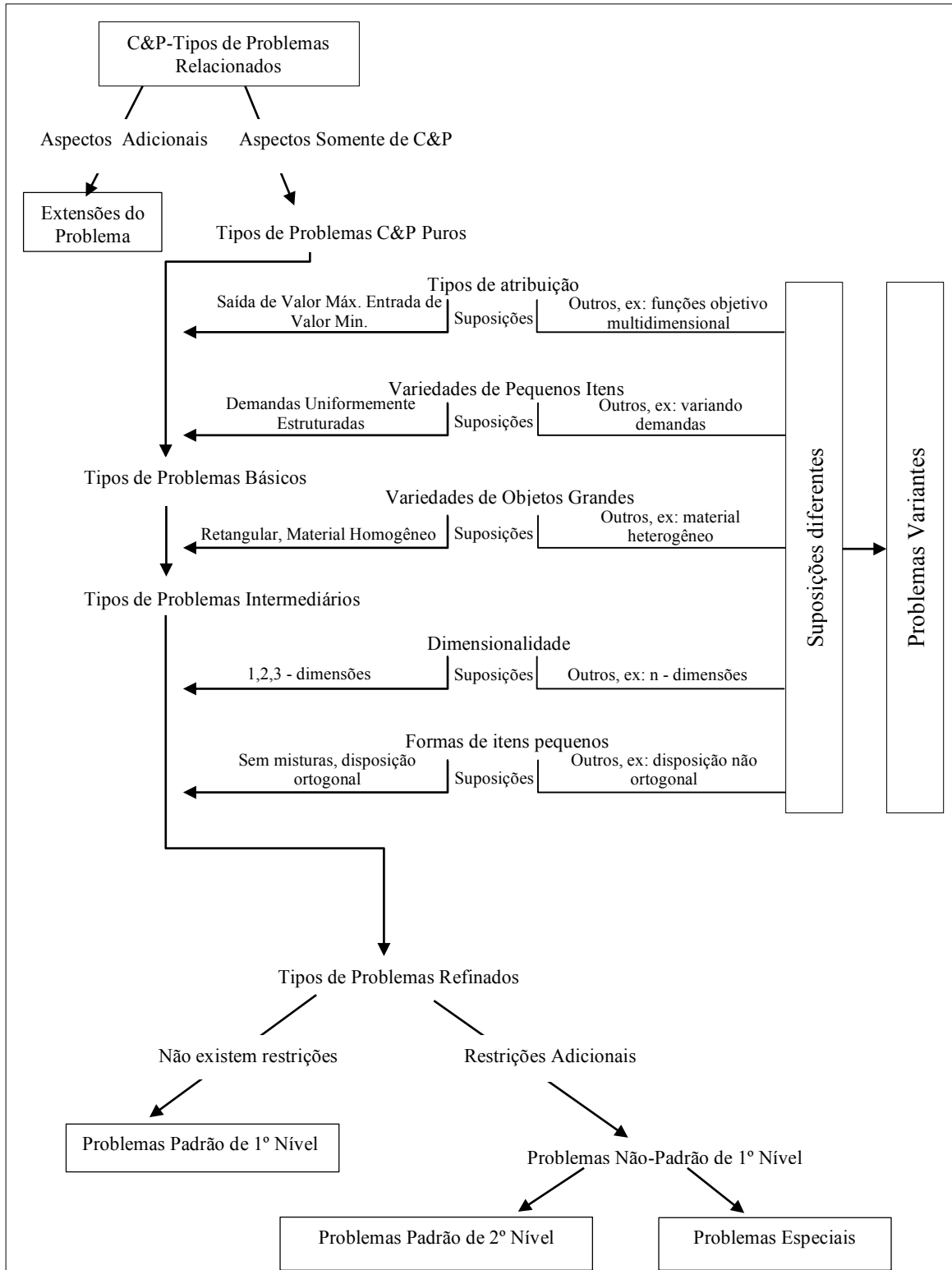
#### 4 Variedade de Peças

- Algumas peças (de diferentes medidas);
- Muitas peças de muitas medidas diferentes;
- Muitas peças de relativamente poucos tipos de medidas diferentes (não congruentes);
- Tipos congruentes.

Segundo Wäscher, Haussner e Schumann (2007), a tipologia sugerida por Dyckhoff (1990) é insuficiente em relação à inclusão dos desenvolvimentos atuais. Portanto, esses autores propõem uma nova classificação, ilustrada, aqui, pela Figura 6. Nela, podem ser observados os problemas existentes, classificados segundo seus objetivos, os quais são: a quantidade de tipos

de peças, a variação na dimensão da placa, a dimensionalidade do problema, a disposição das peças na placa, a quantidade de restrições impostas ao problema e, por fim, a orientação das peças.

**Figura 6 - Visão Geral dos Tipos de Problemas Relacionados ao Corte e ao Empacotamento**



Fonte: Adaptado de Wäscher, Haussner e Schumann (2007).



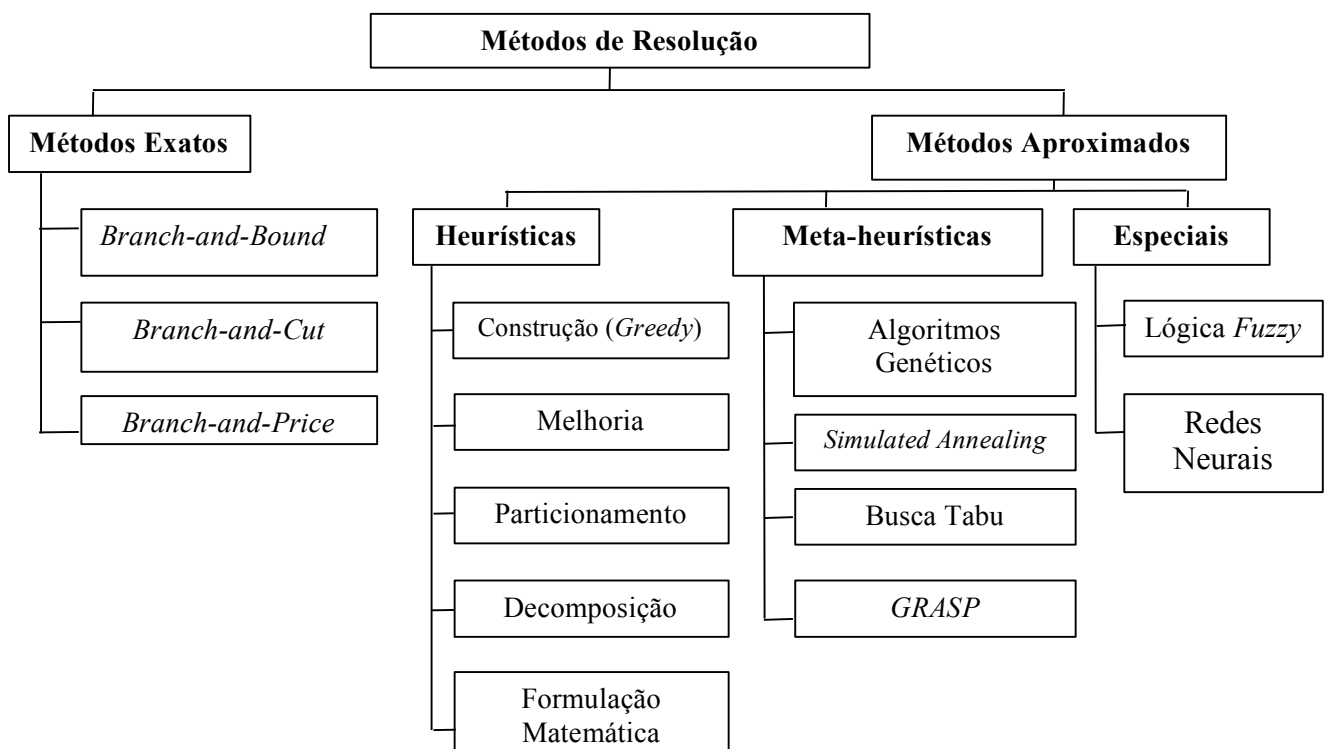
## 2.4 TÉCNICAS DE SOLUÇÃO

Grande parte dos problemas práticos de otimização combinatória são classificados como problemas NP-Completo, problemas NP-*hard* ou NP-Difícil. Segundo Rodrigues (2005), esses problemas não podem ser resolvidos por algoritmos simples em tempo polinomial. Consideram-se como técnicas para a solução de problemas práticos de otimização combinatória os métodos exatos e os métodos aproximados. Os primeiros têm como exemplo o algoritmo de *Branch-and-Bound*, baseado na técnica de dividir o problema em vários outros menores, ou o algoritmo de *Branch-and-Price*, entre outros. Os segundos são constituídos por vários tipos de heurísticas, meta-heurísticas e técnicas especiais para os representar. Entre as heurísticas, podem-se citar a de construção e a gulosa, ambas utilizadas para a resolução de problemas relacionados à otimização combinatória. Entre as meta-heurísticas, têm-se os algoritmos genéticos, busca tabu, *simulated annealing* e GRASP.

Os métodos aproximados, segundo Temponi (2007), não garantem a obtenção da solução ótima do problema, todavia podem se aproximar dela, garantindo soluções subótimas, com baixo esforço computacional, quando comparados à utilização de métodos exatos.

Rodrigues (2005) resume os métodos de resolução dos problemas clássicos de otimização combinatória no organograma como pode ser observado na Figura 7.

**Figura 7 - Métodos de Resolução de Problemas de Otimização**



Fonte: Adaptado de Rodrigues (2005).

O problema de corte bidimensional é considerado NP-*hard*, portanto é complexo de ser resolvido matematicamente e deterministicamente. Por esse motivo são usadas heurísticas e meta-heurísticas como técnicas de solução para o problema de corte bidimensional não guilhotinado, bem como métodos aproximados especiais.

O algoritmo heurístico construtivo consiste em adicionar, geralmente um a um, componentes individuais da solução até encontrar uma solução factível. O mais popular entre os algoritmos heurísticos construtivos é o *greedy* ou “guloso”. O algoritmo heurístico de melhoria se esforça para que, a cada passo, haja uma melhora em sua proposta de solução.

Segundo Temponi (2007) as meta-heurísticas possuem caráter geral, podendo ser aplicadas na resolução de diferentes problemas de otimização, apresentam como principal característica a tentativa de escapar dos ótimos locais, a fim de encontrar um possível ótimo global. Outra definição de meta-heurística vem de Ribeiro (1996), o referido autor define meta-heurística como sendo um procedimento destinado a encontrar uma boa solução, eventualmente a ótima, consistindo na aplicação, em cada passo, de uma heurística subordinada, a qual tem que ser modelada para cada problema específico. Algumas das técnicas classificadas como meta-heurísticas, desenvolvidas e utilizadas nos últimos anos, são: Algoritmos Genéticos, Busca Tabu, Simulated Annealing, GRASP, VNS e outras.

Segundo Rodrigues (2005) as principais meta-heurísticas utilizadas na resolução de problemas clássicos de otimização combinatoria podem ser resumidas em:

- *Simulated Annealing* é uma técnica de busca local probabilística que se baseia no processo de *annealing*, onde um material sólido é aquecido, exposta a altas temperaturas, fazendo com que sua estrutura atômica seja desordenada, em seguida o material é resfriado lentamente, mantendo o quase-equilíbrio térmico, até atingir o estado de mínima energia, momento em que ele assume a estrutura de um cristal perfeito. Assim, o algoritmo de *Simulated Annealing* procura simular um processo equivalente para encontrar a configuração ótima de um problema complexo.

- Busca Tabu é uma meta-heurística que utiliza algoritmo heurístico de busca local combinando estratégias de memória como guia para contornar ou sair de soluções ótimas locais. Esta meta-heurística armazena as últimas transições realizadas por ela na lista tabu, que funciona como uma fila, na qual uma nova transição é adicionada, enquanto a mais antiga é retirada. Com estas informações é possível decidir pela continuidade da exploração do espaço de soluções mesmo na ausência de transições de melhorias da função objetivo.

- Algoritmos Genéticos é uma meta-heurística amplamente utilizada na pesquisa

operacional, esta meta-heurística é baseada no processo de seleção natural de Charles Darwin, em que as características de uma possível solução são medidas por uma função de aptidão, determinando, assim, que os indivíduos mais aptos tenham maiores chances de sobreviver e se reproduzirem. Para reproduzir ou criar uma nova geração, as operações de recombinação, mutação e seleção são usadas. Quando se atinge um critério de parada, o algoritmo termina, e o indivíduo que apresenta as características mais adequadas à solução esperada é selecionado como uma solução ótima ou quase ótima.

- GRASP (*Greedy Randomized Adaptive Search Procedure*) é uma meta-heurística que trabalha com algoritmo heurístico construtivo do tipo guloso, mas que usa uma componente aleatória e adaptativa. É um método iterativo composto por duas etapas: a primeira, na qual são construídas soluções, elemento a elemento. Já na segunda, é realizada uma busca local a fim de determinar um ótimo local na vizinhança de uma solução construída. O processo de seleção das componentes de soluções é baseado em uma função adaptativa gulosa, logo são geradas soluções diferentes em cada interação, em função do grau de aleatoriedade do algoritmo.

- VNS (*Variable Neighborhood Search*), intitulado Busca em Vizinhança Variável, baseia-se na troca sistemática de vizinhanças, partindo de um ponto inicial no espaço de busca, explorando incrementalmente vizinhanças distantes da solução corrente. Parte-se da solução atual para uma nova solução se, e somente se, uma melhora da função objetivo ocorrer. O algoritmo VNS muda a vizinhança como uma forma de sair de soluções ótimas locais. Nesse processo, a solução corrente também é a incumbente, o que não acontece com outras meta-heurísticas. Assim, pode-se afirmar que o algoritmo VNS realiza um conjunto de transições no espaço de busca de um problema e, em cada passo, a transição é realizada para a nova solução incumbente. Se o processo encontra um ótimo local, então o algoritmo VNS muda de vizinhança para sair desse ótimo local e passar para a nova solução incumbente. Como uma consequência dessa estratégia, se o algoritmo VNS encontra o ótimo global, então a busca fica estagnada nesse ponto, sem possibilidade de sair.

## 2.5 ANÁLISE DO ESTADO DA ARTE DO PROBLEMA DE CORTE BIDIMENSIONAL

O problema de corte bidimensional não guilhotinado tem sido classificado como NP-hard, como descrito na seção anterior. Isso ocorre porque problemas mais simples, como o caso do problema de corte unidimensional e o de carregamento de *pallet*, já são considerados NP-hard (LAI; CHAN, 1997). Assim, não existem algoritmos polinomiais para resolver essa classe

de problema.

Segundo Lai e Chan (1997) o problema de corte de estoque é NP-*hard*, por isso é improvável que um método exato e eficiente seja encontrado para resolvê-lo. Métodos exatos podem somente resolver problemas pequenos, e aqueles de porte prático são muitas vezes resolvidos por meio da aplicação de heurísticas para a obtenção de soluções quase ótimas ou aproximadas. Como já mencionado, o problema é difícil e sua complexidade aumenta rapidamente na medida em que aumenta o número de peças disponíveis para o corte. Para gerar uma solução factível e melhorá-la, muitos cálculos são utilizados, e torna-se fácil ficar estagnado em um ótimo local.

Uma discussão completa sobre toda a complexidade, variantes e tipologia do problema em estudo é dada por Dyckhoff (1990), Haessler e Sweeney (1991) e por Wäscher, Haussner e Schumann (2007). Haessler e Sweeney (1991) garantem que, por se tratar de um problema com aplicação variada na indústria, existe um grande investimento para pesquisas na área de problemas de corte e empacotamento, no sentido de encontrar uma solução mais eficaz do que as existentes, comparar soluções alternativas e identificar potenciais benefícios de uma proposta de solução para o problema.

O problema de corte bidimensional possui várias aplicações em indústrias. Por exemplo, Yanasse (1991 apud LAI; CHAN, 1997) apresentou um algoritmo para o problema de corte bidimensional na indústria de madeira; Gemmill (1990 apud LAI; CHAN, 1997) formulou um portfólio que consistia em determinar a melhor combinação de tamanho de peças para armazená-las em estoque; Ferreira et al. (1990 apud LAI; CHAN, 1997) apresentaram uma abordagem heurística para resolver os problemas de corte que surgem na indústria de ferro e aço; Farley (1990) utilizou heurística combinada, com programação inteira, para resolver o problema de corte bidimensional aplicado à indústria de lonas; e Roberts (1984 apud LAI; CHAN, 1997) desenvolveu uma técnica heurística para determinar um cronograma de corte apropriado para bancadas de várias formas e tamanhos, visando o benefício de um fabricante de móveis. Dada a relevância do problema, muito se produz acerca desse assunto. As meta-heurísticas são as mais utilizadas para sua resolução, no entanto não são as únicas técnicas.

Beasley (1985) apresentou um método exato para a resolução do problema de corte bidimensional, que consiste em um procedimento de busca em árvore, no qual foi utilizada uma relaxação lagrangeana com variáveis binárias. O *Simulated annealing* foi utilizado por Dagli e Hajakbari (1990) na busca pela resolução do problema de corte. Os autores utilizaram três métodos para a geração de padrões de corte. O primeiro método usa a heurística para gerar o padrão inicial. O segundo e terceiro métodos empregam a seleção aleatória de padrões, assim

como a alocação aleatória de peças, gerando novos padrões. Também nessa época, Goulimis (1990) propôs um algoritmo de três fases para encontrar a solução ótima do problema de corte, em que, na primeira fase, gera-se o padrão de corte. Na segunda, a solução é associada à programação linear e, na terceira, ela é associada à programação inteira.

Morabito, Arenales e Arcaro (1992) propuseram para a resolução do problema de corte bidimensional guilhotinado o que se chamou de e/ou grafo, uma técnica comumente utilizada em Inteligência Artificial, a qual consiste em um processo de busca considerada como a travessia de um grafo orientado, em que cada nó representa um subproblema, e os arcos representam um relacionamento entre os nós. Para a busca no grafo, os autores combinaram duas estratégias clássicas: as buscas em profundidade e em subida. Além disso, algumas heurísticas foram consideradas. Mais à frente, no ano de 1995, os autores Morabito e Arenales (1995) trabalharam com a mesma proposta de busca e/ou grafo, porém em problemas de corte bidimensional não guilhotinados.

Fayard e Zissimopoulos (1995) apresentaram um algoritmo de aproximação para resolver o problema da mochila bidimensional sem restrições. Primeiramente, o algoritmo empregava uma heurística que seleciona um subconjunto ótimo de peças da solução ótima gerada, resolvendo o problema da mochila unidimensional. Dessa heurística, os autores derivaram um algoritmo de aproximação para que pudessem provar alguns limitantes refinados.

Hadjiconstantinou e Christofides (1995) apresentaram um algoritmo exato para o problema de corte bidimensional ortogonal não guilhotinado, o algoritmo consiste no processo de busca em árvore no qual a dimensão da árvore pesquisada é limitada pelo uso de um limitante derivado do relaxamento de uma formulação de programação dinâmica.

Jakobs (1996) utilizou algoritmo genético para a resolução do problema de empacotamento de polígonos, e o algoritmo foi melhorado pela combinação com métodos determinísticos.

Lai e Chan (1997) apresentaram uma solução para o problema de cortes não guilhotinados com duas ou três dimensões, em que se considerava a minimização de desperdícios utilizando a meta-heurística *simulated annealing*.

Alvarez-Valdes, Parreño e Tamarit (2005) usaram o algoritmo GRASP para propor uma nova solução ao problema de corte bidimensional não guilhotinado e restrito. Na fase construtiva do GRASP, foi utilizado o seguinte critério: novas peças que entrarão no padrão de corte deverão ser escolhidas, levando-se em consideração, primeiramente, o espaço ainda disponível na placa, escolhido pela heurística, e na fase de melhorias, três alternativas distintas foram utilizadas.

Alvarez-Valdes, Parreño e Tamarit (2007) apresentaram um algoritmo que utilizava a meta-heurística busca tabu para a resolução do problema de corte bidimensional não guilhotinado. Vários movimentos baseados na redução e inserção de blocos de peças foram definidos. Procedimentos de intensificação e diversificação, com base na memória de longo prazo, foram incluídos.

Pisinger e Sigurd (2007) propuseram para a resolução do problema de corte e empacotamento, considerando duas dimensões, a técnica de decomposição Dantzig-Wolfe, que divide o problema entre problema principal e subproblema. No principal deles, tratam-se as restrições para a produção do padrão de corte das peças na placa retangular, enquanto no subproblema cortam-se as peças utilizando somente uma única placa. Este é resolvido como um problema de restrição-satisfação, o que torna possível formular algumas restrições adicionais as quais podem ser difíceis de formular utilizando-se modelos de programação inteira mista.

Cintra et. al (2008) utilizaram um algoritmo baseado em programação dinâmica para resolver problemas de corte e empacotamento tridimensional e suas variantes. O algoritmo era baseado em geração de colunas, onde os autores trabalharam com estratégias de melhorias para enfrentar instâncias residuais. Eles também investigaram variantes do problema, nos quais as peças apresentavam tamanhos diferentes, e, finalmente, estudaram o problema do empacotamento bidimensional, no qual também foi utilizado um algoritmo baseado em geração de coluna para a resolução do problema.

Silva, Alvelos e Valério De Carvalho (2010) apresentaram um modelo de programação inteira para o problema de corte com duas e três dimensões. O novo modelo de programação inteira pode ser visto como uma extensão de modelo de corte, proposto por Dyckhoff (1990) para o problema de corte de estoque unidimensional. No modelo proposto, cada variável de decisão era associada ao corte de uma peça, a partir de uma placa ou de uma parte dela, resultante de cortes anteriores.

Del Valle et. al. (2012) propuseram a utilização de algoritmo heurístico, em conjunto com o algoritmo de geração de colunas, para resolver o problema de corte bidimensional com peças irregulares ou polígonos simples. O algoritmo consiste em, inicialmente, embalar a peça com formato irregular em retângulos e, posteriormente, usar os retângulos como peças a serem cortadas na placa.

Alvarez-Valdes, Parreño e Tamarit (2013) apresentaram um outro algoritmo utilizando a meta-heurística GRASP, combinado com a técnica de *path relinking*, para resolver problemas de empacotamento, considerando duas ou três dimensões com múltiplos tamanhos de peças.

Gonçalves e Resende (2013) utilizaram algoritmo genético, baseado em chaves aleatórias (BRKGA), para resolver problemas de corte com duas e três dimensões. O algoritmo BRKGA foi usado pelos autores para construir, passo a passo, a ordem em que as peças eram cortadas nas placas. Além disso, foi usada uma representação de espaço máximo para gerenciar os espaços livres na placa. Os autores utilizaram também duas heurísticas de colocação para determinar o compartimento e o espaço máximo livre onde cada peça seria cortada.

Malaguti, Durán e Toth (2014) utilizaram heurísticas rápidas, modelos de programação linear inteira e o algoritmo de *branch-and-price*, como técnica de solução para o problema de corte bidimensional aplicado na indústria de corte de placas de madeira. Os autores consideraram a maximização da produtividade do equipamento de corte, onde as placas idênticas eram cortadas em paralelo, além de considerarem o objetivo tradicional do problema que é a minimização da perda de matéria-prima, portanto trabalharam com um problema de otimização multiobjetivo.

Russo, Sforza e Sterle (2014) apresentaram um algoritmo de programação dinâmica exato para o problema de corte bidimensional guilhotinado sem restrições em grande escala. Nesse problema, existem pequenas peças retangulares que devem ser cortadas de uma grande placa retangular, sem limites no número de peças. A abordagem de resolução do problema trazida pelos autores é baseada no problema da mochila.

Cui, Cui e Tang (2015) propuseram uma heurística sequencial para resolução do problema de corte e empacotamento bidimensional guilhotinado e não guilhotinado. O algoritmo dos autores produz um padrão de corte a partir de peças cortadas na placa, uma após a outra, até que todas sejam utilizadas. O objetivo da heurística é maximizar o valor do padrão, portanto os autores propuseram um método de correção de valores das peças, que é aplicado após a geração de cada padrão, usando-se uma fórmula de correção de valor.

Dusberger e Raidl (2015) apresentaram uma proposta de solução para o problema de corte e empacotamento bidimensional onde foram empregadas duas abordagens específicas: a de um algoritmo de Busca em Vizinhança Variável e a de uma programação dinâmica. Os autores, a priori, utilizaram a Busca em Vizinhança Variável, com a utilização da estrutura de vizinhança, chamada de ruir-e-reconstruir, na qual partes da solução incumbente são destruídas e reconstruídas, proporcionando, assim, uma busca pela vizinhança. A posteriori, eles aplicaram uma heurística construtiva e programação dinâmica.

Andrade, Birgin e Morabito (2016) propuseram um algoritmo para solucionar uma variante do problema de corte e empacotamento bidimensional guilhotinado, que é o problema de corte bidimensional guilhotinado com sobras utilizáveis. Esse problema pode ser

caracterizado como de natureza de corte residual, pois possibilita disponibilizar, para corte futuro, partes da placa que, em cortes anteriores, foram descartadas. Isso ocorrerá caso essas partes residuais sejam suficientemente grandes para cumprir com as futuras exigências das peças. Nesse problema, as perdas da matéria-prima provocadas pelo corte de cada padrão apresentado não representam necessariamente o desperdício de material. Para a resolução de tal problema, são apresentados dois modelos de programação matemática. Eles, basicamente, consistem em cortar as peças encomendadas, usando-se um conjunto de placas de custo mínimo e, entre todas as soluções possíveis de custo mínimo, escolhendo uma que maximize o valor das sobras geradas em cortes anteriores. Devido às características especiais desses modelos, eles podem ser reformulados como modelos de programação inteira mista.

Ayadi e Barkallah (2016) apresentaram um algoritmo meta-heurístico de otimização por exame de partículas (PSO), adaptado para o problema de corte bidimensional guilhotinado, em conjunto com uma heurística híbrida.



### 3 REPRESENTAÇÃO DE PROPOSTA DE SOLUÇÃO PARA O PROBLEMA DE CORTE BIDIMENSIONAL NÃO GUILHOTINADO

Neste capítulo, o leitor encontrará o conceito de proposta de solução para o problema de corte bidimensional não guilhotinado e como ele pode ser representado, bem como a descrição da representação dessa proposta, a partir do ponto de vista de três trabalhos relevantes na literatura especializada. Ainda neste capítulo, é apresentada uma proposta inédita para a representação de solução do problema de corte bidimensional não guilhotinado.

#### 3.1 VISÃO GERAL

Definir uma proposta de solução para um problema não exige muito esforço, pois está explícito que significa, em outras palavras, identificar uma possível solução para tal. Contudo, a questão que se impõe é como uma proposta de solução pode ser representada. Em geral, sua representação depende do problema e da forma como ele deve ser resolvido.

Existem problemas em que a proposta de solução é representada de maneira direta, ou seja, a forma como a solução está representada pode ser facilmente interpretada por qualquer pessoa, sem a necessidade de algum conhecimento prévio.

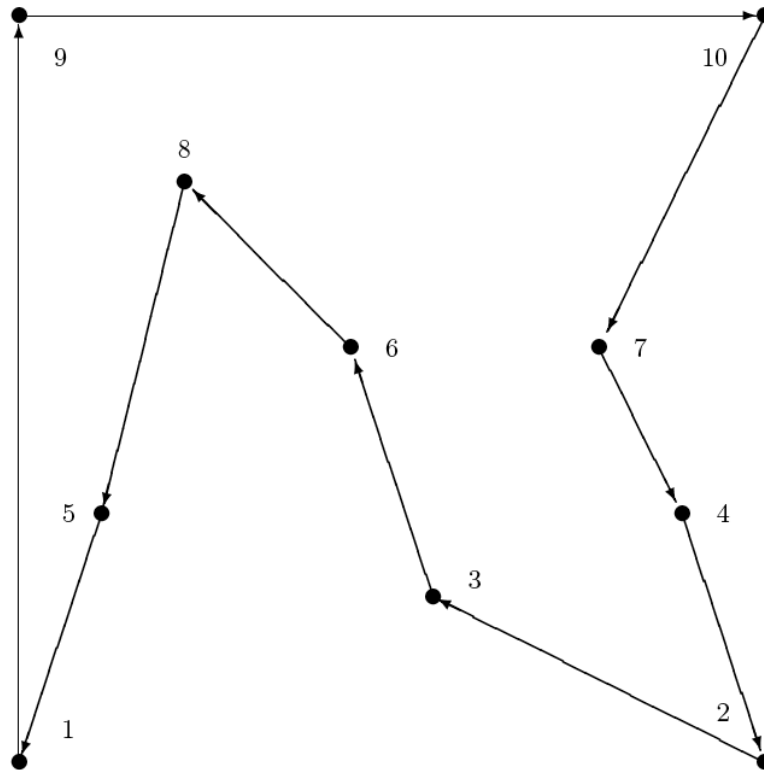
O problema do caixeiro-viajante é um exemplo clássico de representação direta de proposta de solução. Segundo Romero e Mantovani (2004), pode ser enunciado da seguinte forma:

Existe um conjunto de  $n$  cidades,  $V = \{1, 2, 3, \dots, n\}$ , e um conjunto de caminhos (arcos) ligando cada uma das cidades,  $(i, j) \in A$ .  $d_{ij}$  é a distância para ir da cidade  $i$  até a cidade  $j$  em que  $(d_{ij} = d_{ji})$  no caso simétrico. Um caixeiro-viajante deve fazer um *tour* partindo de uma cidade, passando por todas as cidades do conjunto  $V$  uma única vez, e voltando para essa cidade de origem. Pretende-se, assim, encontrar o *tour* de distância mínima.

Na forma mais usada de representação da proposta de solução do problema do caixeiro-viajante, cada elemento do vetor de representação recebe um número que indica uma cidade. Essa representação é realizada por meio de um vetor de dimensão  $n$  (número de cidades), onde as cidades visitadas encontram-se ordenadas de uma forma específica. Os números de 1 a  $n$  identificam a sequência de visitas das cidades. Portanto, esse vetor representa um *tour* em que as cidades são visitadas na mesma sequência em que se encontram no vetor que representa uma proposta de solução. Assim, por exemplo, para o *tour* mostrado na Figura 8, a proposta de solução assume a seguinte forma:

$$p_1 = \begin{array}{|c|c|c|c|c|c|c|c|c|c|} \hline 1 & 9 & 10 & 7 & 4 & 2 & 3 & 6 & 8 & 5 \\ \hline \end{array}$$

**Figura 8** - Exemplo de um *Tour* com 10 Cidades para o Problema do Caixeiro-Viajante



Fonte: Romero e Mantovani (2004).

Verifica-se, assim, que é fácil entender a proposta de solução para o problema do caixeiro-viajante, utilizando-se essa representação direta. Porém, nem todos os problemas são tão fáceis quanto este, visto que existem aqueles, na pesquisa operacional, cuja representação da proposta de solução acontece de maneira codificada e, portanto, é necessário um processo de decodificação (GONÇALVES; RESENDE, 2013).

A representação codificada da proposta de solução indica que para alguém conseguir interpretá-la deverá decodificar a proposta de solução para, então, obter a solução real do problema. O decodificador encontra uma solução que pode ser facilmente interpretada. Cada processo de codificação funciona de uma forma muito específica. O problema de corte bidimensional não guilhotinado é um dos problemas que apresenta sua proposta de solução de maneira codificada.

Para o problema de corte bidimensional não guilhotinado, o decodificador da proposta de solução, normalmente, é um algoritmo heurístico construtivo que, por meio de sua lógica,

constrói passo a passo uma solução para ele. Na literatura especializada, encontraram-se três trabalhos relevantes que apresentam abordagens diferentes para a representação de proposta de solução do problema de corte bidimensional não guilhotinado e da sua decodificação. Essas propostas serão detalhadas nas seções seguintes.

## 3.2 REPRESENTAÇÃO PROPOSTA POR LAI E CHAN

Nesta seção, apresenta-se o algoritmo proposto por Lai e Chan (1997) para a resolução do problema de corte bidimensional não guilhotinado. Os autores trabalham com uma proposta de solução codificada; um decodificador da proposta de solução baseado em um algoritmo heurístico construtivo e uma estrutura de vizinhança baseada na troca da ordem entre duas peças que compõem o vetor ordenado de corte.

### 3.2.1 Representação da Proposta de Solução

Inicialmente, Lai e Chan (1997) propõem trabalhar com um vetor ordenado de corte. Todos os elementos desse vetor são peças que serão cortadas na placa, e a ordem que elas aparecem nas posições do vetor representa a sequência de corte das peças na placa.

O vetor ordenado de corte é arbitrário, seguindo a ordem sequencial das peças. Isso significa que a ordem das peças a serem cortadas, na placa, obedece a uma sequência imposta: primeiro, a peça  $a_1$  é cortada, depois a peça  $a_2$ , e assim sucessivamente. Se existirem  $m$  peças disponíveis para corte, o vetor ordenado de corte começa em sua posição inicial com a peça  $a_1$  e termina em sua posição final com a peça  $a_m$ .

O vetor ordenado de corte das peças, segundo Lai e Chan (1997), é ilustrada na Figura 9.

**Figura 9** - Vetor Ordenado de Corte das Peças na Placa por Lai e Chan (1997)

$$\text{Vetor ordenado de corte} = [a_1, a_2, a_3, a_4, \dots, a_m]$$

Fonte: Elaborado pela autora.

É preciso destacar que, mesmo seguindo a sequência de corte das peças, baseando-se em um vetor ordenado de corte, diferentes posicionamentos dessas peças, na placa, geram diferentes padrões de corte. Esse vetor poderia ser encarado como uma proposta de solução

codificada para o problema de corte bidimensional não guilhotinado, pois, se interpretado por pessoas diferentes que não tenham conhecimento do algoritmo de decodificação da proposta de solução, poderá produzir configurações de corte diferentes. Cada pessoa pode interpretar o corte de cada peça em locais diferentes dentro da placa, mesmo se estiver seguindo uma sequência de peças idênticas. Por isso, o decodificador torna-se tão importante nesse problema.

O decodificador é um algoritmo que, a cada passo, estabelece qual peça será cortada na placa, a localização, dentro da placa, onde a peça será cortada, e quais retângulos livres que o corte da peça produziu. Tem-se como resultado desse processo decodificador um conjunto de informações chamado por Lai e Chan (1997) de padrão de corte. Esse conjunto de informações é composto por: vetor ordenado de corte, onde são identificadas quais as peças que realmente serão cortadas na placa e sua sequência de corte; lista de retângulos livres produzidos pelas peças cortadas na placa; valor objetivo ou *fitness* do desperdício de área da placa, após o corte das peças que constam no vetor ordenado de corte.

O padrão de corte é a solução real do problema, no qual se tem sempre exatamente a mesma configuração de corte, independentemente se a proposta de solução foi interpretada por uma pessoa, por outra, ou por um computador. Porém, a proposta de solução de Lai e Chan (1997) pode ser considerada uma proposta codificada, porque precisa de um algoritmo heurístico construtivo, considerado um algoritmo decodificador. Para que uma pessoa ou computador consiga chegar à solução real do problema em estudo, não basta ter o padrão de corte proposto por Lai e Chan (1997), será preciso ter necessariamente o conhecimento das regras e especificidades do algoritmo decodificador. Portanto, todas as vezes que um padrão de corte (vetor ordenado de peças, lista de retângulos livres e valor objetivo do desperdício da placa) for interpretado, será necessário processá-lo em conjunto o algoritmo decodificador, pois, sem ele, como já escrito anteriormente, continua-se correndo o risco de o padrão de corte ser interpretado de formas divergentes, tanto por pessoas quanto máquinas diferentes.

A representação do padrão de corte proposto como solução codificada por Lai e Chan (1997) é ilustrada na Tabela 1.

**Tabela 1** - Representação do Padrão de Corte Proposto como Solução Codificada por Lai e Chan (1997)

Vetor ordenado de corte de peças na placa	01, 02, 03, 04, 06, 10
Lista com dimensões dos retângulos livres	Relativo à primeira peça 01 ( $R_1, R_2$ ) Relativo à segunda peça 02 ( $R_2, R_3$ ) Relativo à terceira peça 03 ( $R_2, R_4$ ) Relativo à quarta peça 04 ( $R_4, R_5, R_6$ ) Relativo à quinta peça 06 ( $R_4, R_5, R_7, R_8$ ) Relativo à sexta peça 010 ( $R_4, R_5, R_7, R_9, R_{10}$ )
Valor Objetivo do Desperdício da placa	6 unidades de área da placa

Fonte: Elaborado pela autora

A seguir, o processo construtivo/decodificador proposto por Lai e Chan (1997) é detalhado.

### 3.2.2 Algoritmo Heurístico Construtivo

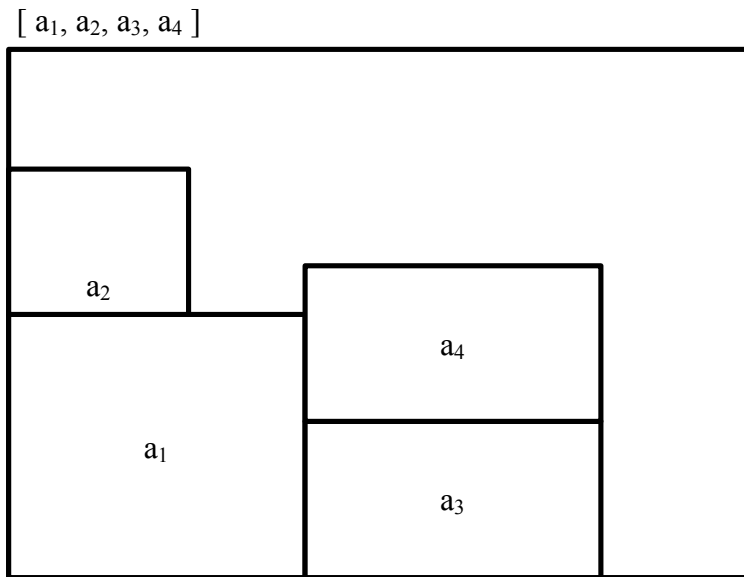
O decodificador é baseado em um algoritmo heurístico construtivo que corta as peças, uma a uma, na placa.

O problema de corte abordado por Lai e Chan (1997) é o do tipo bidimensional não guilhotinado, com uma diversidade de peças com tamanhos diferentes. Para cada padrão de corte gerado, obtêm-se a ordem de corte das peças, os retângulos livres gerados pelas peças cortadas na placa e o desperdício de área da placa relacionado ao padrão.

Diferentes posições das peças, dentro da ordem de corte, formam diferentes padrões de corte. Pode-se usar o método de troca de quaisquer duas peças cortadas na ordem de corte para gerar todos os vizinhos desses padrões. Esse processo pode ser estendido para gerar todo o espaço de corte, ou seja, todos os possíveis arranjos desses padrões.

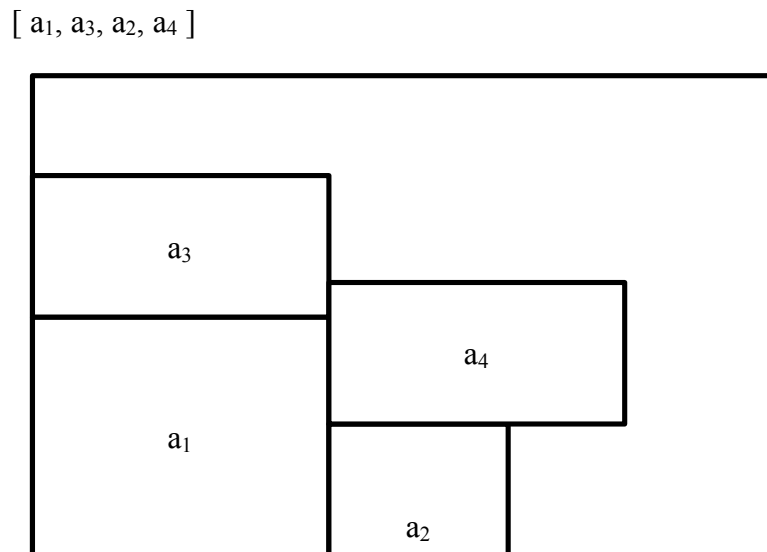
Como exemplo de geração de vizinhos, têm-se as Figuras 10 e 11. Nelas, a troca na ordem das peças  $a_2$  e  $a_3$  não causa nenhuma diferença no desperdício de área da placa, pois não mudaram as peças, mas, sim, a ordem delas. No entanto, essa troca na ordem das peças irá afetar a disponibilidade de espaço que pode ser usado para cortar uma nova peça. Portanto, assume-se que a área total de todas as peças disponíveis para corte deve ser maior do que a área da placa.

**Figura 10** - O Padrão de Corte é Ilustrado pela Ordem de Corte [a<sub>1</sub>, a<sub>2</sub>, a<sub>3</sub>, a<sub>4</sub>]



Fonte: Elaborado pela autora.

**Figura 11** - O Padrão de Corte é Ilustrado pela Ordem de Corte [a<sub>1</sub>, a<sub>3</sub>, a<sub>2</sub>, a<sub>4</sub>]



Fonte: Elaborado pela autora.

Lai e Chan (1997) propõem o seguinte algoritmo heurístico construtivo para tratar o problema de corte bidimensional não guilhotinado:

(1) Geração dos retângulos livres. Depois de cortada cada peça dentro da placa, é necessário mapear a placa para saber quais são os retângulos livres para as próximas peças a serem cortadas.

(2) Seleção de qual retângulo livre será utilizado para o corte de uma peça.

(3) Geração do padrão de corte. Após um processo construtivo, onde peça a peça é colocada na placa, em uma certa ordem, para ser cortada, tem-se um padrão de corte.

A seguir, serão detalhados os três passos encontrados no algoritmo heurístico construtivo de Lai e Chan (1997).

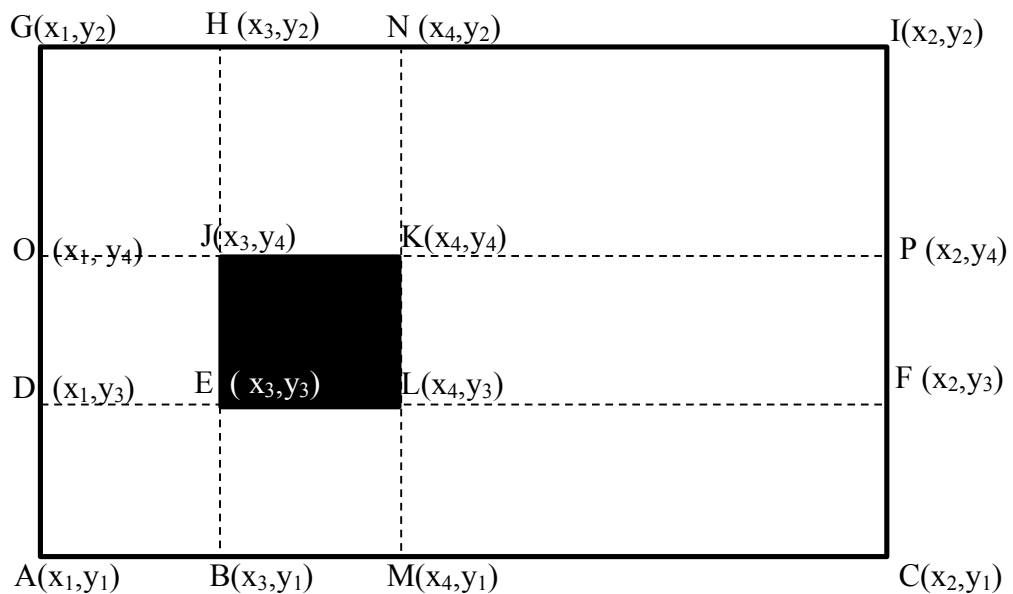
### 3.2.2.1 Geração dos Retângulos Livres

O passo de geração de retângulos livres é usado para gerar as próximas áreas de corte depois que uma peça é cortada na placa. Portanto, esse passo mapeia a placa após o corte de uma peça, verificando quais foram os retângulos livres deixados na placa para o corte das próximas peças.

O mapeamento da placa em busca de retângulos livres é realizado verificando retângulos livres à esquerda da peça que foi cortada, logo após é verificada a possibilidade de retângulos livres à direita da peça cortada. Na sequência, verifica-se a existência de retângulos livres abaixo e, por último, acima da peça cortada na placa. Para formular um problema clássico de corte de placa, Lai e Chan (1997) aplicaram a técnica de álgebra intervalar.

Como mostra a Figura 12, foi assumido que  $x_1 \leq x_3 \leq x_4 \leq x_2$  e  $y_1 \leq y_3 \leq y_4 \leq y_2$ .

**Figura 12** - Processo de Diferença na Geração do Intervalo



Fonte: Elaborado pela autora.

Usando a álgebra de intervalos, Lai e Chan (1997) descrevem o processo de diferença da seguinte maneira:

Diferença:  $[(x_1, y_1), (x_2, y_2)] - [(x_3, y_3), (x_4, y_4)]$ .

=

$[(x_1, y_1), (x_3, y_2)],$

$[(x_4, y_1), (x_2, y_2)],$

$[(x_1, y_1), (x_2, y_3)],$

$[(x_1, y_4), (x_2, y_2)]]$

Ou seja,

Diferença:  $[A, I] - [E, K] = [[A, H], [M, I], [A, F], [O, I]]$

Resumindo o que se mostra acima, se a placa possui coordenadas  $(x_1, y_1) = (0, 0)$  em seu canto inferior esquerdo, e  $(x_2, y_2) = (100, 70)$  em seu canto superior direito, contudo ainda não tem peças cortadas em seu interior, isso significa que o retângulo livre, nesse momento, apresenta as mesmas dimensões que a placa. Ao cortar uma peça com as coordenadas  $(x_3, y_3) = (20, 20)$ , em seu canto inferior esquerdo, e com  $(x_4, y_4) = (40, 40)$ , em seu canto superior direito, há uma subtração, na placa, do espaço ocupado pela peça. Então, os retângulos livres, existentes na placa, devem ser mapeados. O procedimento para mapear retângulos livres, como já citado anteriormente, faz-se a partir da verificação dos retângulos livres à esquerda da peça que foi cortada, logo após, à direita, na sequência abaixo e, por último, acima da peça cortada na placa. Geram-se, portanto, os retângulos livres a seguir, com as primeiras coordenadas correspondentes ao canto inferior esquerdo do retângulo livre, seguido das coordenadas do canto superior direito do retângulo livre: o retângulo livre à esquerda da peça cortada tem coordenadas  $(x_1, y_1) = (0, 0)$ ,  $(x_3, y_2) = (20, 70)$ ; o retângulo livre à direita da peça cortada tem coordenadas  $(x_4, y_1) = (40, 0)$ ,  $(x_2, y_2) = (100, 70)$ ; o retângulo livre abaixo da peça cortada tem coordenadas  $(x_1, y_1) = (0, 0)$ ,  $(x_2, y_3) = (100, 20)$ ; por último, o retângulo livre acima da peça cortada tem coordenadas  $(x_1, y_4) = (0, 40)$ ,  $(x_2, y_2) = (100, 70)$ .

### 3.2.2.1.1 Processo de Diferença

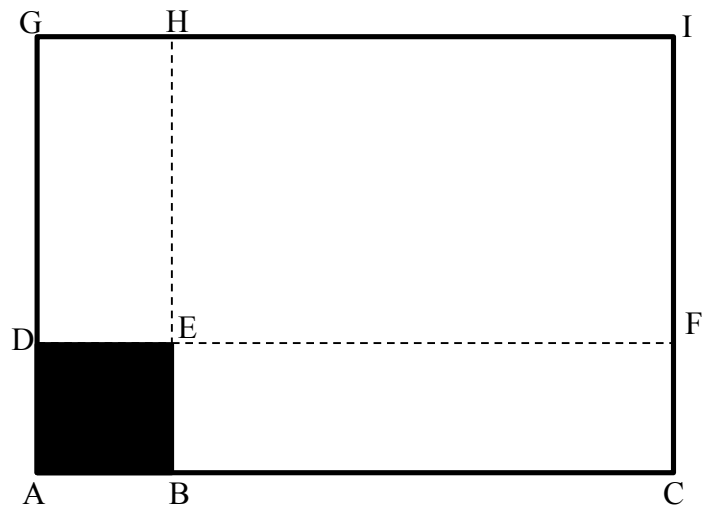
O processo de diferença, segundo Lai e Chan (1997), envolve a geração de possíveis retângulos livres após ser cortada uma peça em uma placa. Utilizando-se o exemplo de corte bidimensional não guilhotinado, ilustrado na Figura 13(a), suponha que uma peça  $[A, E]$  seja cortada em uma placa  $[A, I]$ . Lembrando-se de que o mapeamento da placa, em busca de retângulos livres, é realizado verificando-se retângulos livres à esquerda da peça que foi



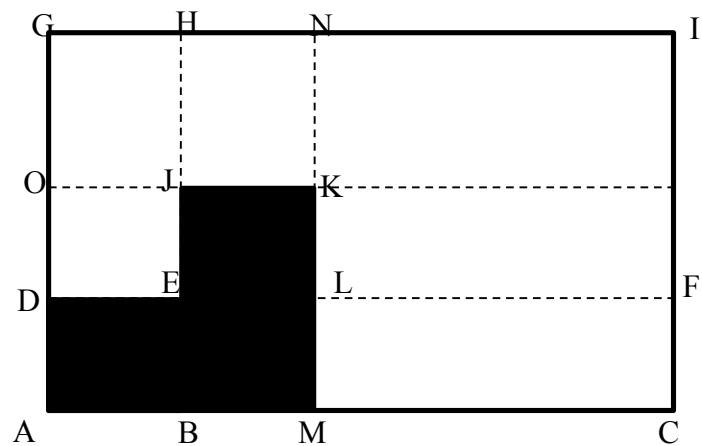
cortada. Na sequência, à direita da peça cortada, depois, abaixo, e, por último, acima da peça cortada na placa. Portanto, a lista de retângulos livres  $[[A, G], [B, I], [A, C], [D, I]]$  é gerada após o processo de diferença. Tomando como ilustração o exemplo da Figura 13(b), observa-se que outra peça  $[B, K]$  é cortada na placa utilizando o retângulo livre  $[B, I]$ . A lista de retângulos livres gerados ao final do processo de diferença é  $[[B, H], [M, I], [B, C], [J, I]]$ , e o processo de diferença de  $[D, I] - [B, K]$  gera a lista de retângulos livres  $[[D, H], [L, I], [D, F], [O, I]]$ .

Portanto, ao final do corte das peças  $[A, E]$  e  $[B, K]$  na placa  $[A, I]$ , tem-se a lista de retângulos livres  $[[A, G], [A, C], [B, H], [M, I], [B, C], [J, I], [D, H], [L, I], [D, F], [O, I]]$ . O processo de diferença continua a gerar uma nova lista de retângulos livres depois que outra peça for cortada na placa.

**Figura 13** - Processo de Eliminação dos Intervalos Gerados



(a) Primeira peça cortada na placa e seus retângulos livres



(b) Segunda peça cortada na placa e seus retângulos livres

### 3.2.2.1.2 Processo de Eliminação

Após a lista de retângulos livres ter sido gerada pelo processo de diferença, alguns retângulos devem ser eliminados para poupar espaço de armazenamento na memória do computador. Retângulos sem uma das dimensões ou aqueles que estão totalmente contidos em outro retângulo livre são removidos da lista.

Primeiramente, executa-se o processo de verificação cruzada, como foi chamado por Lai e Chan (1997). Nesse processo, são comparadas as coordenadas de cada retângulo livre da lista de retângulos livres, após o processo de diferença, para verificar e eliminar dessa lista os retângulos livres que estão contidos em outros retângulos com as mesmas características, de maior ou igual dimensão. O segundo processo de eliminação foi chamado por Lai e Chan (1997) de autoeliminação, onde os retângulos livres serão analisados e eliminados caso suas coordenadas se mostrem representando uma única dimensão. A seguir, têm-se exemplos dos dois processos de eliminação descritos por Lai e Chan (1997):

1. Verificação Cruzada: compara cada retângulo livre com os outros retângulos livres da lista para ver se ele está totalmente contido em outro retângulo.

Eliminar  $[L, I]$  de  
 $[[A, G], [A, C], [B, H], [M, I], [B, C], [J, I], [D, H], [L, I], [D, F], [O, I]]$  se  
 $([M, I] > [L, I])$ .

2. Autoeliminação: verifica se o retângulo livre é formado por uma única dimensão ou não.

Eliminar  $[B, H]$  da lista de retângulos livres se  $[B, H]$  representar somente uma reta.

Ao final do processo de diferença e eliminação, a lista de retângulos livres resultante da ilustração vista nas Figuras 13(a) e 13(b) seria  $[[M, I], [D, H], [O, I]]$ . Os retângulos livres, na lista, representam a disponibilidade de escolhas para a próxima peça a ser cortada no espaço restante da placa.

### **3.2.2.2 Seleção do Retângulo Livre**

Segundo Lai e Chan (1997), durante o passo de geração de retângulos livres, geraram-se vários retângulos livres. Em um próximo passo, um deles deve ser escolhido para cortar uma peça. Nesse segundo passo, é necessário algum critério para selecionar qual retângulo livre será

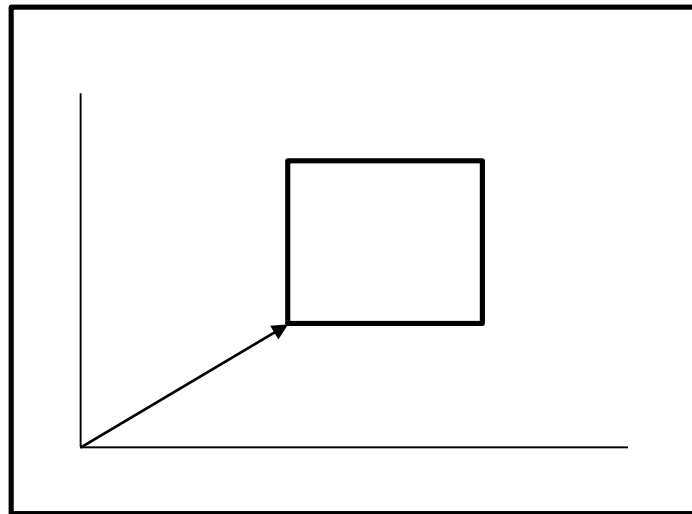
usado no próximo processo de corte. Um dos critérios de seleção deve ser baseado no tamanho. Desse modo, o retângulo livre deve ser grande o suficiente para acomodar a próxima peça a ser cortada em seu interior. Além disso, precisa-se de uma estratégia de seleção para escolher um deles, dentre os vários retângulos livres resultantes do primeiro critério de seleção, principalmente nas primeiras iterações do algoritmo heurístico construtivo.

Lai e Chan (1997) usaram uma heurística para essa finalidade, a qual foi denominada "a mais compacta possível". Ela consiste em selecionar, dentre os retângulos livres resultantes da primeira busca, aquele que possui a menor distância entre a origem da placa e o canto inferior esquerdo do retângulo livre. Em outras palavras, as peças devem ser colocadas o mais próximo possível do canto inferior esquerdo da placa, como ilustra a Figura 14(a).

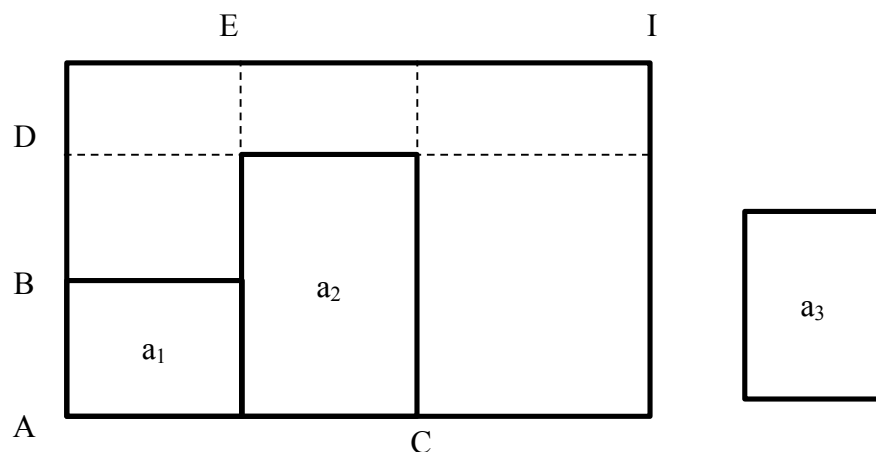
Suponha que as peças  $a_1$  e  $a_2$  foram posicionadas na placa, e três retângulos livres foram gerados  $[B, E]$ ,  $[C, I]$  e  $[D, I]$ . A peça  $a_3$  está aguardando para ser posicionada na placa, como mostra a Figura 14(b).

A peça  $a_3$  não pode ser cortada no retângulo livre  $[D, I]$  em virtude de seu tamanho. Assim, a peça  $a_3$  pode ser somente cortada nos retângulos livres  $[B, E]$  e  $[C, I]$ . Baseado na heurística "o mais compacto possível", a peça deveria ser cortada no retângulo livre  $[B, E]$ , porque a distância entre  $A$  e  $C$  é maior do que a distância entre  $A$  e  $B$ .

**Figura 14 - Seleção de Retângulo Livre**



*(a) Distância da Origem.*



*(b) A peça  $a_3$  está aguardando para ser cortada na placa.*

Fonte: Elaborado pela autora.

### **3.2.2.3 Geração do Padrão de Corte**

Baseado nos procedimentos de Geração dos Retângulos Livres e de Seleção do Retângulo Livre, Lai e Chan (1997) puderam gerar um padrão de corte. Este representa uma solução real para o problema e, de acordo com eles, é composto por: vetor ordenado de corte, o qual traz a sequência das peças cortadas na placa, deixando explícito, inclusive, que nem todas as peças disponíveis foram cortadas e, portanto, nem todas elas constam no vetor ordenado de corte; lista de retângulos livres, que resulta do corte das peças que se encontram no vetor ordenado de corte e que, para chegar a essa lista, precisa seguir as regras de colocação das peças na placa descritas nos procedimentos de Geração dos Retângulos livres e Seleção do Retângulo

Livre. Por fim, o padrão de corte de Lai e Chan (1997) também é composto pelo valor objetivo ou *fitness* do desperdício de área da placa depois de cortadas todas as peças que se encontravam no vetor ordenado de corte, ou seja, aquele espaço que sobrou na placa, entretanto não pode ser reaproveitado para o corte de nenhuma outra peça.

Portanto, sabendo-se do algoritmo heurístico construtivo ou decodificador da proposta de solução do problema de corte bidimensional não guilhotinado e aplicando suas regras na construção do padrão de corte, independentemente de quem faça o processo construtivo ou a decodificação, o resultado do padrão de corte será o mesmo.

### 3.2.3 Definição de Estrutura de Vizinhaça

Uma solução inicial é gerada aleatoriamente, em seguida ela é decodificada, a partir da utilização o AHC. Essa solução transforma-se na solução corrente e, para esta, é gerada uma proposta de solução vizinha, a qual é decodificada usando-se novamente o AHC. Essa solução vizinha pode ou não ser aceita, de acordo com a estratégia do algoritmo *simulated annealing*, elaborado por Lai e Chan (1997). Ao final do algoritmo proposto por Lai e Chan (1997), tem-se um padrão de corte composto por vetor ordenado de corte, lista de retângulos livres e valor do desperdício da placa. Basicamente, a estrutura de vizinhaça consiste na troca da ordem de duas peças dentro do vetor ordenado de corte, cada troca levará a um padrão de corte diferente ou a um padrão de corte vizinho, pois, com esse movimento, não se altera somente a ordem das peças em um vetor, mas, sim, a ordem de corte dessas peças na placa, alterando suas listas de retângulos livres e o desperdício da placa.

Lai e Chan (1997) definem que a possibilidade de troca entre duas peças no vetor ordenado de corte deve ocorrer de acordo com o valor âncora. Tal valor é um número real, gerado aleatoriamente, e pode variar de dois até a quantidade de peças que compõem o vetor ordenado de corte. O valor gerado deve ser arredondado para o número inteiro mais próximo. Este controla o grau de alterações do padrão de corte, portanto, com ele, determina-se quantas peças, a partir da última delas contida no vetor ordenado de corte, estão disponíveis para a troca.

Uma vez determinadas quantas peças estarão disponíveis para a troca, aleatoriamente, escolhem-se duas que trocarão de lugar, ou seja, que modificarão sua ordem dentro do vetor ordenado de corte, alterando todo o seu padrão, a partir da primeira peça que foi trocada. Com a troca de uma peça por outra, a lista de retângulos livres altera-se e, por consequência desse ato, as peças que vinham na sequência, dentro do vetor ordenado de corte, podem não mais se encaixar nesses novos retângulos livres. Isso dá a oportunidade para que outras peças, que antes

não faziam parte do vetor ordenado de corte, entrarem no vetor, gerando, dessa forma, uma enorme mudança no padrão de corte de modo geral. Assim, tem-se um padrão de corte vizinho.

Após as trocas, o desperdício da placa pode manter-se igual, ou seja, as peças mudaram de lugar dentro da placa, entretanto a somatória de suas áreas continua a mesma. Pode ocorrer também o aumento no desperdício. Nesse caso, a troca da ordem das peças na placa faz com que alguma delas, que cabia antes, não caiba mais, aumentando o desperdício em um primeiro momento. Porém, em ambos os casos, houve uma movimentação das peças no padrão de corte, gerando novos retângulos livres. Sendo assim, pode-se colocar o algoritmo heurístico construtivo em prática novamente para que este tente alocar peças que antes não faziam parte do vetor ordenado de corte por falta de espaço na placa e agora, com as movimentações, tornam-se possíveis novas inserções.

Por exemplo,

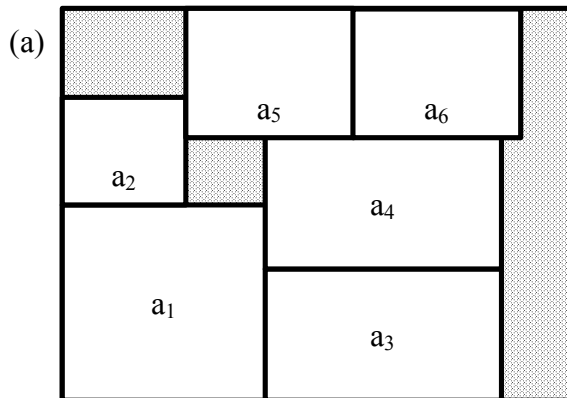
$$[a_1, a_2, a_3, a_4, a_5, a_6]$$

←  
Valor âncora igual a cinco.

No exemplo anterior, o valor âncora é cinco. Isso significa que podem ser selecionadas aleatoriamente duas peças do vetor ordenado de corte entre  $a_2$  e  $a_6$  para futura troca. Quanto maior for o valor âncora, maior é a possibilidade de uma grande diferença entre os padrões de corte vizinhos. Por exemplo, veja a Figura 15 (a), nesta ilustração a troca entre as peças  $a_3$  e  $a_6$  provocaram mudança no padrão de corte atual gerando um vizinho, pois ao trocar a ordem de corte de  $a_3$  para  $a_6$ , os cortes das peças  $a_4$  e  $a_5$  foram impactados, resultando em falta de espaço livre para que a peça  $a_3$  pudesse ser cortada na placa, logo a peça  $a_3$  ficou fora do padrão de corte vizinho.

Se o valor âncora fosse dois, isso significaria que somente as peças  $a_5$  e  $a_6$  poderiam ser usadas para a troca, veja a Figura 15 (b), nesta ilustração é possível verificar que o padrão de corte mudou, porém os impactos provocados pela troca entre as duas peças foi menor em relação ao exemplo anterior da Figura 15(a).

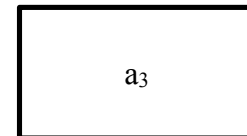
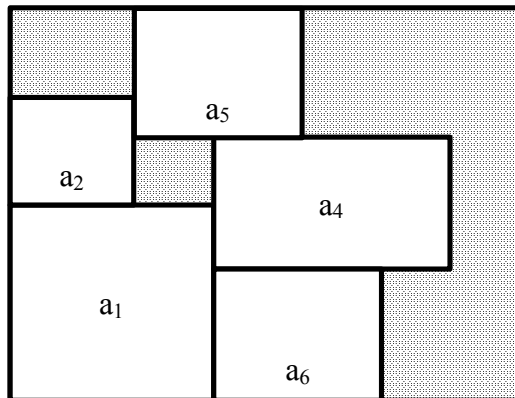
**Figura 15** - Estrutura de Vizinhança. (a) Se o Valor Âncora é 5, Quaisquer Dois Elementos Entre  $a_2$  e  $a_6$  Podem Ser Seleccionados para a Troca. (b) Se o Valor Âncora é 2, Apenas os Elementos Entre  $a_5$  e  $a_6$  Podem Ser Seleccionados para a Troca



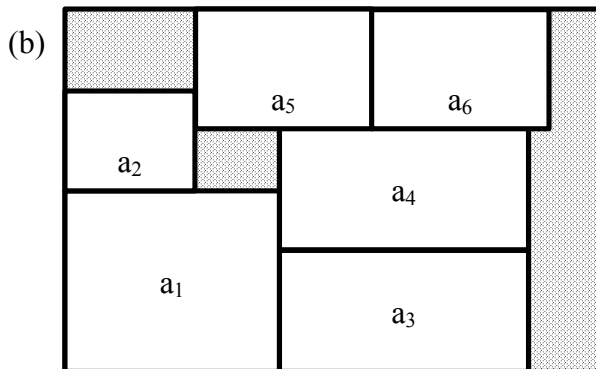
$[a_1, a_2, a_3, a_4, a_5, a_6]$

Se o valor âncora é 5, duas peças entre  $a_2$  e  $a_6$  podem ser seleccionadas para a troca.

Suponha a troca de  $a_3$  e  $a_6$ .

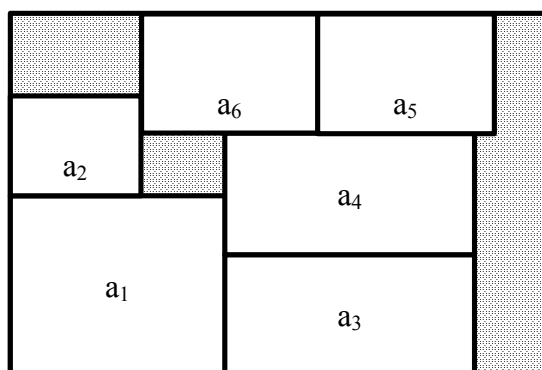


Resultado da troca:  $a_3$  não pode ser cortado na placa.



$[a_1, a_2, a_3, a_4, a_5, a_6]$

Se o valor âncora é 2, somente peças entre  $a_5$  e  $a_6$  podem ser seleccionadas para a troca.



### 3.3 REPRESENTAÇÃO PROPOSTA POR ALVAREZ-VALDES, PARREÑO E TAMARIT

Nesta seção, apresenta-se o algoritmo proposto por Alvarez-Valdes, Parreño e Tamarit (2007) para a resolução do problema de corte bidimensional não guilhotinado com restrições. Os autores trabalham com uma proposta de solução codificada; um decodificador baseado em um algoritmo heurístico construtivo e uma estrutura de vizinhança baseada na retirada e na inserção de blocos de peças.

#### 3.3.1 Representação da Proposta de Solução

A proposta de solução do problema de corte bidimensional não guilhotinado apresentada por Alvarez-Valdes, Parreño e Tamarit (2007) também é codificada, como ocorre com Lai e Chan (1997), descrito na seção 3.2. Inicialmente, os autores contam com as peças que estão disponíveis para o corte e a placa onde as peças serão cortadas. A partir dessas informações, o algoritmo heurístico construtivo que, nesse caso, também age como um decodificador de propostas de solução para o problema e começa a construir uma proposta de solução.

A representação da proposta de solução do problema de corte bidimensional não guilhotinado, apresentada por Alvarez-Valdes, Parreño e Tamarit (2007), é composta por quatro informações: uma lista de blocos de peças cortadas na placa, que obedece à sequência de cortes; uma lista com a quantidade de peças que formam cada bloco; uma lista de retângulos livres produzidos com o corte dos blocos e o valor total das peças cortadas na placa. Esse valor pode ser convertido em desperdício da placa quando considerado que as peças têm o mesmo valor de sua área.

As quatro informações representam uma proposta de solução para o problema de corte bidimensional não guilhotinado. Todavia, essa representação é codificada e necessita de um decodificador para que as informações que compõem a proposta de solução tornem-se a resolução do problema original, ou seja, um padrão de corte.

A representação da proposta de solução codificada de Alvarez-Valdes, Parreño e Tamarit (2007) é ilustrada na Tabela 2, na qual é possível encontrar a lista de blocos como primeira informação que compõe tal representação. Essa lista indica a sequência em que peças  $i$  são cortadas na placa, onde  $i$  vai de 1 a  $m$ . No entanto, antes de serem cortadas, as peças  $i$  são agrupadas, formando um bloco de peças do mesmo tipo  $i$ , como se fossem uma grande peça. No exemplo da Tabela 2, tem-se, como primeiro bloco cortado na placa, o bloco das peças do tipo 1. Após esse bloco, tem-se o bloco das peças do tipo 2, depois o bloco das peças do tipo



4, 3, 7, 10 e, por fim, o bloco das peças do tipo 11. Na segunda linha da Tabela 2, encontra-se a quantidade de cópias das peças que foram agrupadas e que deram origem aos blocos de cada tipo de peça  $i$ . No exemplo da Tabela 2, o bloco das peças do tipo 1 foi composto por 6 exemplares da peça 1. Já no bloco das peças do tipo 2, foram utilizadas 4 cópias da peça 2, e assim sucessivamente. A terceira informação encontrada na Tabela 2 é a lista de retângulos livres que restaram na placa após cortar todos os blocos da lista de blocos. A lista de retângulos livres traz as dimensões desses retângulos vazios. E, por fim, na última linha da Tabela 2, encontra-se a soma dos valores de todas as peças que foram cortadas na placa.

**Tabela 2** - Representação da Proposta de Solução Codificada por Alvarez-Valdes, Parreño e Tamarit (2007)

Lista ordenada de blocos	1, 2, 4, 3, 7, 10, 11
Lista de quantidade de peças por bloco	6, 4, 3, 4, 2, 1, 1
Lista de retângulos livres	03x55 38x05 16x05 05x01 30x10 03x45 38x05
Valor Total das peças cortadas na placa	194

Fonte: Elaborado pela autora

A representação da proposta de solução é codificada, porque se as quatro informações que representam uma proposta de solução para o problema forem entregues a um funcionário de uma empresa que trabalha com corte de placa, este não teria condições de interpretá-las para gerar um padrão de corte que seria aplicado à placa. Por esse motivo, é necessário que um decodificador atue junto a essa proposta.

O decodificador é o próprio algoritmo heurístico construtivo proposto por Alvarez-Valdes, Parreño e Tamarit (2007), que, passo a passo, estabelece a ordem dos blocos de peças que serão cortados na placa e em qual lugar da placa elas serão cortados. Ao final desse algoritmo decodificador, vê-se que ele interpretou as quatro informações que compõem a representação da proposta de solução de Alvarez-Valdes, Parreño e Tamarit (2007), exemplificadas na Tabela 2, gerando um padrão de corte que é a solução real para o problema de corte bidimensional não guilhotinado. No entanto, fica claro que sem o algoritmo decodificador não seria possível a uma pessoa ou a um computador chegar sempre ao mesmo resultado de padrão de corte, pois ambos não iriam conhecer as regras desse algoritmo para cortar as peças na placa. Portanto, todas as vezes que for preciso interpretar a representação da

proposta de solução de Alvarez-Valdes, Parreño e Tamarit (2007), será necessário executar o algoritmo decodificador, já que a proposta de solução está codificada.

A seguir será detalhado o processo de construção de uma proposta de solução para o problema de corte bidimensional não guilhotinado, por meio do algoritmo heurístico construtivo de Alvarez-Valdes, Parreño e Tamarit (2007).

### 3.3.2 Algoritmo Heurístico Construtivo

Segundo Alvarez-Valdes, Parreño e Tamarit (2007), a construção de uma solução é um processo iterativo em que se combinam três elementos, a saber: uma lista  $P$ , de peças  $i$  disponíveis para corte na placa. Inicialmente,  $P$  é formada pela lista completa de peças  $i$ , onde  $i$  começa em 1 e vai até  $m$  (quantidade de tipos de peças); a lista  $S$ , formada pelas peças já cortadas na placa que, a priori, encontra-se vazia; a lista  $RL$ , formada por retângulos livres, na qual uma peça  $i$  pode ser cortada, de início, contendo apenas a placa  $R$ , que é composta por comprimento  $C$  e altura  $A$ . A cada passo, um retângulo livre é escolhido de  $RL$ . Da lista de peças  $P$ , uma peça  $i$  é escolhida para ser cortada, e a lista  $S$  é atualizada com a peça que foi cortada na placa. Isso, normalmente, produz novos retângulos livres que farão parte de  $RL$ , e o processo se repete até que  $RL$  seja igual a zero ( $RL = 0$ ), ou até que nenhuma das peças  $i$ , restantes em  $P$ , encaixem-se em um dos retângulos livres restantes em  $RL$ . A seguir, apresentam-se os passos do algoritmo heurístico construtivo.

#### ***Passo 0 – Inicialização:***

$RL = \{R\}$ ,  $RL$  é o conjunto de retângulos livres que, inicialmente, é composto por toda a placa  $R$ , ou seja, um grande retângulo livre.

$P = \{p_1, p_2, \dots, p_m\}$ ,  $P$  é o conjunto de peças disponíveis para corte.

$S = \emptyset$ ,  $S$  é o conjunto de peças cortadas que, a princípio, é vazio, pois não há peças cortadas na placa. À medida que as peças forem cortadas na placa, aquelas utilizadas no corte serão inseridas nesse conjunto.

Peças do mesmo tipo podem aparecer agrupadas em blocos retangulares.

O conjunto  $P$  é, de início, ordenado de acordo com três critérios:

1. Ordenar  $P$  de modo decrescente, dando prioridade às peças com restrição em seu limitante inferior, ou seja, as peças que apresentarem um número mínimo de cópias ( $P_i$ ) a serem cortadas na placa e que obtiverem o maior valor como resultado da multiplicação do número mínimo de cópias pela área da peça serão melhor classificadas na lista  $P$ . ( $P_i * c_i * a_i$ ).

2. Se existir um empate, por exemplo, se várias peças ou, até mesmo, todas as peças  $i$  tiverem seu número mínimo de cópias  $P_i$  igual ao valor zero ( $P_i = 0, \forall i$ ), deve-se ordenar  $P$  de maneira decrescente, seguindo outro critério: o da razão entre o valor da peça e sua área ( $v_i/(c_i * a_i)$ ), ou seja, as peças mais lucrativas serão melhor classificadas.
3. Se novamente existir um empate, por exemplo, se várias peças ou, até mesmo, todas as peças  $i$  tiverem seu valor igual à sua área ( $v_i = c_i * a_i, \forall i$ ), deve-se ordenar  $P$  de maneira decrescente, seguindo outro critério: o critério da área, isto é, as peças  $i$  que obtiverem a maior área ( $c_i * a_i$ ) serão melhor classificadas.

***Passo 1 – Escolhendo o retângulo livre:***

Escolha um  $RL^*$ , sendo  $RL^*$  um candidato a participar da iteração corrente como o menor retângulo livre escolhido de  $RL$ , em que uma peça  $i \in P$  pode caber e sendo  $p_i$  a melhor peça classificada e disponível em  $P$  na iteração corrente. Se existir um empate em relação ao  $RL^*$  escolhido, o critério de desempate é escolher aquele  $RL^*$  com menor distância de qualquer canto da placa. A razão para essa decisão é tentar satisfazer a demanda de pequenas peças com pequenos retângulos livres, deixando os grandes para peças grandes. Alvarez-Valdes, Parreño e Tamarit (2005) detalham que se for escolhido um retângulo grande, no início, e usá-lo para cortar uma peça pequena nele, os retângulos livres, resultantes do corte, podem ser inúteis para as peças grandes a serem cortadas posteriormente.

Se tal  $RL^*$  não existir, pare, pois isso significa que não existem mais retângulos livres onde as peças restantes em  $P$  possam ser cortadas.

Se existir, siga para o Passo 2.

***Passo 2 – Escolhendo a peça para cortar:***

Escolha a peça  $i$  e o número de cópias ( $x_i$ ) dessa mesma peça, disponíveis para corte. A quantidade de cópias da peça  $i$  não pode ultrapassar o seu limitante superior,  $Q_i$ , que indica o número máximo de peças  $i$  disponíveis para corte. ( $x_i \leq Q_i$ ). A peça  $i$  é escolhida seguindo a ordem da lista  $P$ . Essas peças de mesmo tipo serão agrupadas formando um bloco chamado de  $B^*$  a ser cortado no canto de  $RL^*$ , que está mais próximo de um canto da placa.  $B^*$  é, portanto, um bloco candidato a ser cortado na placa, na iteração corrente. O total da área do bloco  $B^*$  deve ser menor que a área do  $RL^*$ .

O bloco  $B^*$  a ser cortado, na maioria dos casos, não preenche completamente o retângulo  $RL^*$ .  $B^*$  é movido para o canto mais próximo da placa. Se for esse o caso, ele sai do retângulo

$RL^*$  no qual foi cortado e entra, total ou parcialmente, em outro retângulo livre caso ele se encontre mais próximo do canto da placa. Dessa forma, as regiões livres do padrão de corte estão concentradas em direção ao centro e podem ser mais facilmente incorporadas ao Passo 3.

Deve-se atualizar  $P, S, RL$  e  $Q_i$ .

Atualiza-se  $S$  com o tipo da peça  $i$  e o número  $x_i$  de cópias da peça cortada. Faça  $Q_i = Q_i - x_i$ . Se  $Q_i = 0$ , retire a peça  $i$  da lista  $P$ .

**Passo 3 – Atualizando a lista  $RL$ :**

Adicionar em  $RL$  os retângulos livres produzidos quando cortado  $B^*$  de  $RL^*$ . Deve-se levar em conta as possíveis mudanças em  $RL$  quando se move o bloco  $B^*$ . A menos que o bloco  $B^*$  se encaixe exatamente no retângulo  $RL^*$  e ele não se mova.

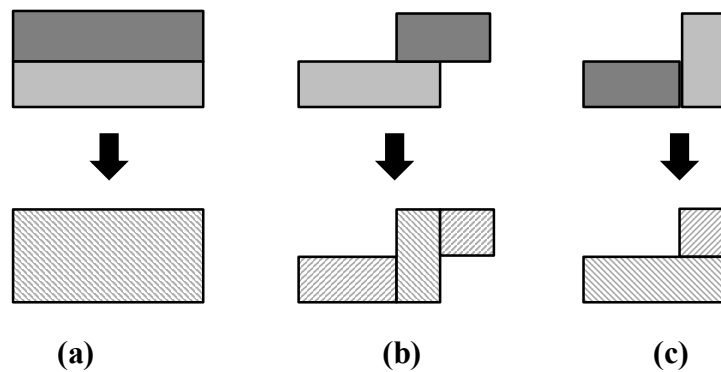
Mesclar os retângulos livres para favorecer o corte de novas peças de  $P$ .

Voltar ao Passo 1.

Segundo Alvarez-Valdes, Parreño e Tamarit (2007), apesar de manter uma lista de retângulos livres  $RL$ , tem-se um espaço livre irregular e poligonal, em que as peças disponíveis para corte podem ser cortadas. Uma maneira de adaptar a lista  $RL$  para a flexibilidade do corte não guilhotinado é mesclar alguns dos retângulos livres da lista, produzindo novos retângulos dessa natureza, para que as peças disponíveis para corte se encaixem melhor.

Quando se mesclam dois retângulos livres, no máximo três novos retângulos podem aparecer, geralmente um grande retângulo e dois pequenos, como ilustra a Figura 16. Entre as várias alternativas para mesclar, Alvarez-Valdes, Parreño e Tamarit (2007) tentaram selecionar a melhor, ou seja, aquela que possibilita cortar as peças de melhor qualidade, classificadas na lista ordenada  $P$ . Com esse objetivo, Alvarez-Valdes, Parreño e Tamarit (2007) estabeleceram algumas condições:

1. Se a peça que se encaixa no retângulo maior for de melhor qualidade que as peças que se encaixam nos retângulos originais, mesclar.
2. Se a peça que se encaixa no retângulo grande tiver a mesma qualidade que as peças que se encaixam nos originais, mesclar. Isso se a área do retângulo grande for maior do que a área de cada retângulo original.
3. Se a peça que se encaixa no retângulo grande possuir uma qualidade pior que as peças que se encaixam nos retângulos originais, não mesclar.

**Figura 16** - Mesclar Dois Retângulos Livres

Fonte: Alvarez-Valdes, Parreño e Tamarit (2007).

Na Figura 16, representam-se vários casos possíveis, sendo que, nesta, os dois retângulos originais serão sempre mesclados. Na Figura 16 (a), o novo retângulo é maior do que os retângulos originais, e todas as peças encaixadas nestes caberão no primeiro. Na Figura 16 (b), os novos retângulos não são maiores do que os originais. Eles serão mesclados somente se o novo retângulo central acomodar peças de melhor qualidade do que aquelas encaixadas nos retângulos originais. Na Figura 16 (c), um dos novos retângulos é maior do que os originais e, portanto, eles serão mesclados, a menos que a peça encaixada no retângulo vertical original seja de melhor qualidade em comparação às peças que se encaixam nos novos retângulos.

Ao final do processo construtivo, uma proposta de solução é encontrada e composta por: uma lista de blocos  $S$ , nessa lista são encontrados, em ordem de corte, os blocos de peças cortadas; uma lista com a quantidade de cópias de cada peça que forma um bloco; uma lista de retângulos livres  $RL$  e o valor total  $\sum_i v_i x_i$  das peças cortadas na placa.

### 3.3.3 Definição de Estrutura de Vizinhaça

A solução inicial do problema é obtida por meio da aplicação do algoritmo heurístico construtivo, descrito anteriormente. Porém, a partir da solução inicial, pode-se procurar por soluções vizinhas melhores.

Segundo a proposta de Alvarez-Valdes, Parreño e Tamarit (2007), deve-se trabalhar com dois tipos de estrutura de vizinhaça: retirada de bloco e inserção de bloco. Na retirada de bloco, o tamanho de um bloco existente é reduzido, retirando algumas de suas linhas e colunas, ou o bloco pode ser totalmente extinto, principalmente quando todas as linhas e colunas desse bloco são retiradas. Na inserção de bloco, um novo bloco é adicionado à solução.

Para ambas as transições, a seguir, serão apresentados os seguintes itens: um esquema dos procedimentos e um exemplo detalhado.

### 3.3.3.1 Retirada de Bloco

#### ***Passo 0 - Inicialização:***

$S$  = Lista de blocos da solução corrente.

$RL$  = Lista de retângulos livres.

$P$  = Conjunto de peças disponíveis para corte.

#### ***Passo 1 - Escolhendo o bloco a retirar:***

Escolher um bloco  $B$ , candidato a sair da placa, sendo  $B$  um dos blocos de  $S$  com  $k$  colunas e  $l$  linhas, formado por  $x_i$  cópias da peça  $i$ .

Selecionar o número  $r$  de colunas ou linhas para eliminar,  $1 \leq r \leq k$  ou  $1 \leq r \leq l$ , respeitando a restrição do limitante inferior ( $P_i$ ), onde não se deve excluir totalmente o bloco de peças  $i$ . Se essa peça  $i$  tem seu limitante inferior maior que zero. Se  $P_i = 0$ , o bloco pode ser excluído completamente.

O bloco retirado de  $S$  faz com que  $P$  e  $RL$  sejam atualizados.

#### ***Passo 2 - Mova os blocos restantes para os cantos mais próximos:***

A lista de retângulos livres  $RL$  é, conseqüentemente, atualizada.

#### ***Passo 3 - Preencha os retângulos livres com novos blocos:***

Aplicar o algoritmo heurístico construtivo novamente.

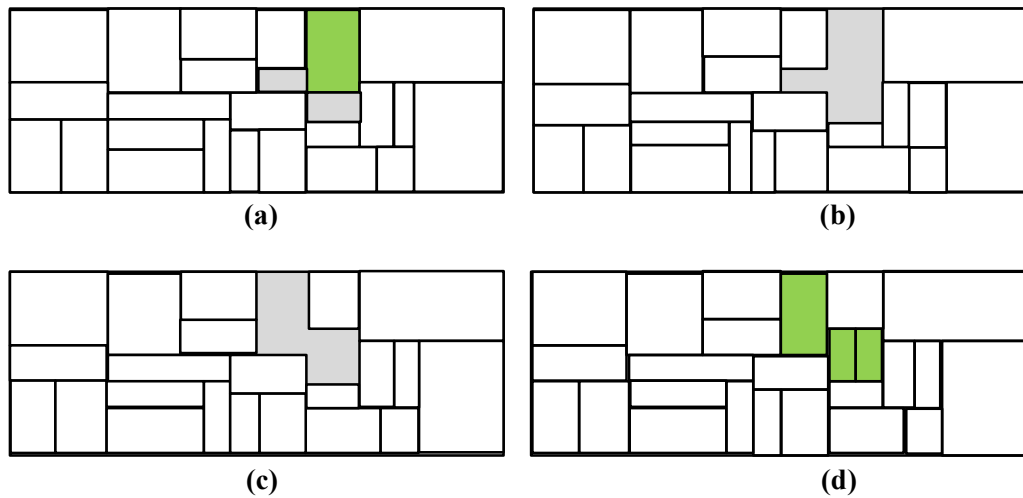
O algoritmo é iniciado a partir das listas correntes  $RL$ ,  $S$  e  $P$ . Antes de prosseguir com a construção, retângulos livres em  $RL$  passam pelo procedimento de mesclar retângulos a fim de melhor acomodar as peças de  $P$ .

Ao selecionar a peça a ser cortada, a peça ou o bloco eliminado no Passo 1 não é considerado(a) disponível para ser utilizado(a) novamente. Torna-se proibido(a) até que outra peça ou bloco tenha sido incluído(a) na solução modificada.

#### ***Passo 4 - Mesclar os blocos com a mesma estrutura:***

Alvarez-Valdes, Parreño e Tamarit (2007) propõem mesclar blocos de uma mesma peça com o mesmo comprimento e altura caso estejam adjacentes e tenham um lado comum, ou se um deles puder ser movido para fazer os dois retângulos ficarem adjacentes ao mesmo lado.

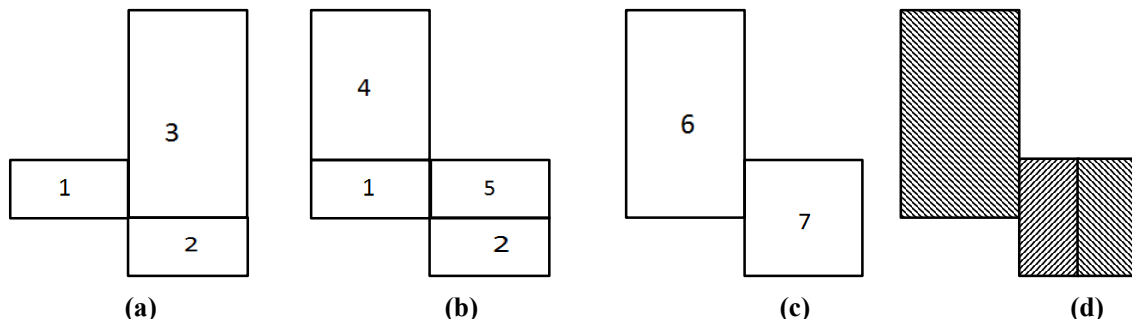
**Figura 17** - Retirada de Blocos. (a) Seleção, (b) Retirada, (c) Transição para o Canto e (d) Preencher



Fonte: Alvarez-Valdes, Parreño e Tamarit (2007).

Na Figura 17, apresenta-se um exemplo de uma transição de retirada (ALVAREZ-VALDES; PARREÑO; TAMARIT, 2007). A placa é  $R = (120 \times 45)$ , sua origem  $(0 \times 0)$  encontra-se no canto inferior esquerdo da placa. Os tipos de peças disponíveis vão de 1 até  $m$ , com  $m = 22$ , e a quantidade de peças disponíveis para corte na placa é de 25 ( $M = 25$ ). Isso significa que algum dos 22 tipos de peças tem mais de um exemplar disponível para corte. Na Figura 17 (a), apresenta-se uma solução com 23 peças que não consegue acomodar 2 peças de tamanho  $(6 \times 12)$ . O conjunto  $RL$  de retângulos livres é composto por  $RL_1 = (60, 24, 72, 30)$ , onde os dois primeiros números representam o canto inferior esquerdo do retângulo livre  $RL_1$ , e os dois últimos números representam o canto superior direito de  $RL_1$ , e  $RL_2 = (72, 18, 84, 24)$  (em cinza claro). No Passo 1, um bloco composto de uma peça  $(12 \times 21)$  (em verde) é selecionado para ser retirado e, portanto, ele desaparece da solução, criando um novo retângulo livre  $RL_3 = (72, 24, 84, 45)$  que é adicionado a  $RL$  (Figura 17(b)). No Passo 2, um bloco composto de uma peça  $(12 \times 15)$  é movido para o canto direito de cima. Portanto,  $RL = \{RL_1, RL_2, RL_4, RL_5\}$ , onde  $RL_4 = ((60, 30), (72, 45))$  e  $RL_5 = ((72, 24), (84, 30))$  (Figura 17(c)). No Passo 3, o processo construtivo preenche os retângulos livres. Primeiro,  $RL_1$  e  $RL_4$  são mesclados, formando  $RL_6 = ((60, 24), (72, 45))$ , e  $RL_2$  e  $RL_5$  são mesclados, formando  $RL_7 = ((72, 18), (84, 30))$ . Então, o  $RL_7$  é selecionado e as duas peças  $(6 \times 12)$  são cortadas nele, preenchendo-o completamente. Finalmente,  $RL_6$  é selecionado e nele a peça inicialmente retirada é cortada. A solução final, ótima, é ilustrada na Figura 17(d). A evolução dos retângulos livres pode ser vista mais claramente na Figura 18.

**Figura 18** - Retângulos Livres na Transição de Retirada. (a) Passo 1, (b) Passo 2, (c) Mesclar e (d) Preencher



Fonte: Alvarez-Valdes, Parreño e Tamarit (2007).

### 3.3.3.2 Inserção de Bloco

#### ***Passo 0 - Inicialização:***

$S$  = Lista de blocos.

$RL$  = Lista de retângulos livres.

$P$  = Conjunto de peças disponíveis para corte.

#### ***Passo 1 - Escolhendo o bloco a inserir:***

Escolher um tipo de peça  $i$ , cuja quantidade de cópias  $x_i$  seja menor que seu limitante superior,  $Q_i$ , formando um bloco de peças do tipo  $i$ , com  $k$  colunas e  $l$  linhas.

#### ***Passo 2 - Selecione a posição para inserir o novo bloco.***

#### ***Passo 3 - Remova as peças da solução corrente que foram cobertas pelo novo bloco:***

Atualizar  $S$  e  $P$ , alguns blocos originais foram retirados parcialmente ou, ainda, eliminados totalmente, e essas peças devem voltar a ser disponibilizadas na lista  $P$ .

Atualizar  $RL$ , novos retângulos livres podem aparecer com a eliminação ou retirada dos blocos no passo 3.

#### ***Passo 4 – Preencha os retângulos livres com blocos novos.***

#### ***Passo 5 – Mescle os blocos que tenham a mesma estrutura.***

Passos 4 e 5 são iguais ao processo de Retirada de Bloco.

No Passo 2, uma estratégia com dois passos consecutivos foi considerada por Alvarez-Valdes, Parreño e Tamarit (2007) para selecionar a posição do novo bloco. Em ambos os passos, o bloco é posicionado parcial ou totalmente, cobrindo um ou mais retângulos livres, como ilustra a Figura 19.

- No primeiro passo da estratégia, para cada retângulo livre consideram-se as quatro alternativas em que um canto do retângulo é escolhido para um canto

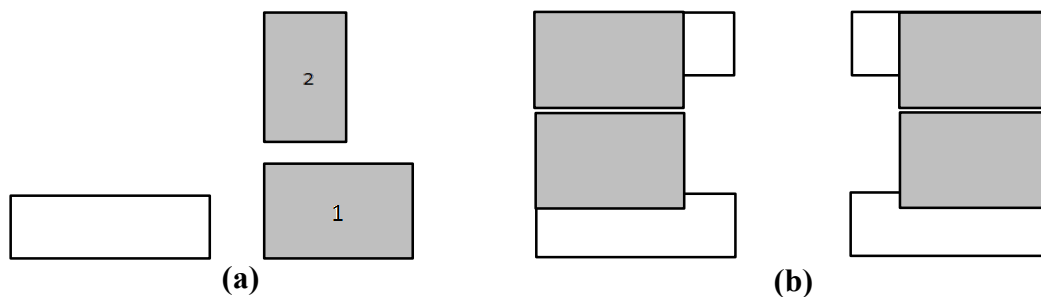


correspondente de um bloco. Se as dimensões do bloco são maiores do que as do retângulo, o bloco pode sobrepor-se com os outros blocos da solução corrente ou pode ocupar uma parte de outros retângulos livres.

- No segundo passo, dentre todos os retângulos livres analisados no passo anterior, deve-se selecionar somente um retângulo livre que produza a maior intersecção com o bloco, se o canto inferior esquerdo do bloco foi colocado no canto inferior esquerdo do retângulo. Para esse retângulo livre, as quatro alternativas descritas acima são consideradas.

Na Figura 19, Alvarez-Valdes, Parreño e Tamarit (2007) ilustram um exemplo do segundo passo. Na Figura 19(a), dois retângulos livres, em cinza, são considerados para acomodar o novo bloco, em branco. Faz-se a intersecção entre o novo bloco e todos os retângulos livres disponíveis, e a maior área comum corresponde ao retângulo 1, então esse retângulo é escolhido. Na Figura 19(b), os quatro cantos do retângulo livre são considerados para posicionar o canto do novo bloco.

**Figura 19** - Selecionando a Posição do Novo Bloco. (a) Maior Área Comum e (b) Posições Possíveis

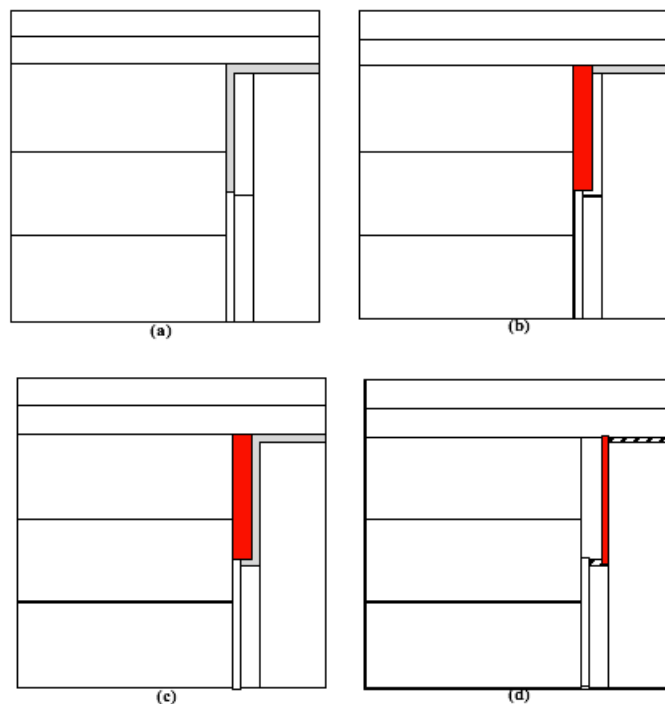


Fonte: Alvarez-Valdes, Parreño e Tamarit (2007).

Na Figura 20, Alvarez-Valdes, Parreño e Tamarit (2007) exibem um exemplo de uma transição de inserção. A placa é  $R = (100 \times 100)$ , os tipos de peças disponíveis vão de 1 até  $m$ , com  $m = 15$ , e a quantidade de peças disponíveis para corte na placa é de 50 peças ( $M = 50$ ). Isso significa que alguns dos 15 tipos de peças têm mais de um exemplar disponível para corte. A Figura 20(a) mostra uma solução de valor  $z = 27539$ . O conjunto  $RL$  de retângulos livres é composto por  $RL_1 = ((70, 41), (72, 81))$  e  $RL_2 = ((72, 80), (100, 81))$ . No passo 1, selecionou-se uma peça  $i = 5$  de dimensões  $(6 \times 40)$  com  $Q_i = 5$  e somente duas cópias na solução corrente. Considerou-se a formação de um bloco  $B$ , composto por apenas uma peça. No Passo 2, posicionou-se  $B$  sobre  $RL_1$ , por se tratar do retângulo livre que proporcionou maior intersecção com a área do novo bloco, seguindo a estratégia do passo 2.

Foi selecionado o canto superior esquerdo do retângulo para acomodar o canto superior esquerdo do bloco.  $B$  cobre completamente  $RL_1$  e parte do  $RL_2$ , que se torna  $RL_3 = ((76, 80), (100, 81))$  (Figura 20(b)). Portanto, no Passo 3, removem-se as peças da solução inicial que foram cobertas por  $B$ . Isso produz dois novos retângulos livres  $RL_4 = ((76, 40), (78, 80))$  e  $RL_5 = ((72, 40), (76, 41))$  (Figura 20(c)). No Passo 4, a lista  $RL$  é formada por  $\{RL_3, RL_4, RL_5\}$ . Primeiro,  $RL_3$  e  $RL_4$  são mesclados, produzindo  $RL_6 = ((76, 40), (78, 81))$  e  $RL_7 = ((78, 80), (100, 81))$ . Embora nenhuma das peças restantes possam caber tanto em  $RL_3$  quanto em  $RL_4$ , uma peça  $i = 13$  de dimensões  $(2 \times 41)$  agora se encaixa em  $RL_6$ . A nova solução é melhor do que a inicial e tem um valor  $z^* = 27718$ , ideal para o problema (Figura 20(d)).

**Figura 20** - Inserção de Blocos. (a) Inicial, (b) Inserir, (c) Eliminar Sobreposição e (d) Mesclar e Preencher



Fonte: Alvarez-Valdes, Parreño e Tamarit (2007).

Segundo Alvarez-Valdes, Parreño e Tamarit (2007), a cada iteração, estudam-se todos os vizinhos possíveis aplicando à solução corrente as transições de retirada e de inserção.

- *Retirada*
  1. Para cada bloco da solução corrente, considerar todas as possibilidades de retirada nas direções adjacentes aos retângulos livres.
- *Inserção*

1. Considerar todos os possíveis blocos que podem ser construídos a partir das peças que o número de cópias na solução,  $x_i$ , seja menor que  $Q_i$ .

2. Considerar todas as alternativas para colocar o bloco em um retângulo livre.

Alvarez-Valdes, Parreño e Tamarit (2007) também adotaram uma estratégia para atributos proibidos. A estratégia consiste em fazer com que a peça eliminada, ao selecionar a peça a ser cortada na atual iteração, não seja considerada disponível para ser reutilizada, até que outra peça tenha sido incluída na solução modificada. Com relação ao critério de aspiração, os autores não adotaram essa estratégia em seu algoritmo.

### 3.4 REPRESENTAÇÃO PROPOSTA POR GONÇALVES E RESENDE

Nesta seção, apresenta-se o algoritmo proposto por Gonçalves e Resende (2013) para a resolução do problema de corte bidimensional não guilhotinado com restrições. Os autores apresentam uma proposta de solução codificada; um decodificador da proposta de solução e uma estrutura de vizinhança baseada em chaves aleatórias viciadas.

#### 3.4.1 Representação da Proposta de Solução

A representação da proposta de solução do problema de corte bidimensional não guilhotinado apresentada por Gonçalves e Resende (2013) é composta por um vetor com três parcelas. A primeira delas representa a sequência de corte das peças na placa (*Box Packing Sequence* - BPS), a segunda parcela representa heurísticas de colocação de cada peça na placa (*Vector of Placement Heuristics* - VPH) e a terceira representa a orientação de cada peça (*Vector of Box Orientations* - VBO) que foi cortada na placa. Unindo as informações encontradas nas três parcelas desse vetor, tem-se um padrão de corte das peças na placa como solução para o problema original. Portanto, ao final do processo de resolução do problema, tem-se uma resposta para o problema real, em que a representação dessa solução do problema foi decodificada. Todavia, mesmo obtendo-se a resposta decodificada como resposta real para o problema, se o padrão de corte é dado a uma pessoa ou máquina para que uma delas o execute, terá que processar, em conjunto com as três parcelas do vetor padrão de corte, o algoritmo decodificador, pois sem o conhecimento das regras que ele impõe para a escolha de retângulos livres e corte das peças não é possível reproduzir sempre a mesma configuração de corte das peças na placa.

De acordo com Gonçalves e Resende (2013), inicialmente, a proposta de solução é codificada e representada por um vetor contendo três parcelas de números reais, gerados aleatoriamente no intervalo  $[0, 1]$ . Diante disso, cada proposta de solução codificada é composta de  $3n$  variáveis, como ilustrado na Figura 21. As primeiras  $n$  variáveis são usadas para obter a sequência de corte de peças, as variáveis  $n + 1$  até  $2n$  são usadas para obter o valor das heurísticas de colocação, e as variáveis  $2n + 1$  até  $3n$  são usadas para obter o vetor de orientação das peças.

A Representação da Proposta de Solução codificada de Gonçalves e Resende (2013) é ilustrada na Figura 21.

**Figura 21** - Representação da Proposta de Solução Codificada por Gonçalves e Resende (2013)

$$\text{Proposta Solução Codificada} = (p_1, p_2, \dots, p_n, h_{n+1}, h_{n+2}, \dots, h_{2n}, o_{2n+1}, o_{2n+2}, \dots, o_{3n})$$

$p_1, p_2, \dots, p_n$	$h_{n+1}, h_{n+2}, \dots, h_{2n}$	$o_{2n+1}, o_{2n+2}, \dots, o_{3n}$
Sequência de Peças Cortadas	Heurística de Colocação	Orientação das Peças

Fonte: Elaborado pela autora.

Como já mencionado anteriormente, esse vetor representa uma proposta de solução codificada e, portanto, necessita de um decodificador para indicar exatamente onde cada peça deve ser cortada na placa. Esse decodificador gerará um padrão de corte das peças.

Ao receber o padrão de corte, o responsável por cortar as peças na placa precisará executar, em conjunto com esse padrão, o algoritmo decodificador para saber o local exato onde deverá cortar cada peça. As informações contidas no padrão de corte não são suficientes para que se saiba exatamente onde cada peça será cortada na placa, visto que existe sempre uma dependência do algoritmo decodificador, mesmo tendo o padrão de corte decodificado. O processo de decodificação usa como entrada a proposta de solução codificada, decodifica essa proposta, segundo um algoritmo determinístico proposto por Gonçalves e Resende (2013), e tem como resultado final um padrão de corte que é a solução real do problema de otimização combinatória para o qual um valor objetivo ou *fitness* pode ser calculado.

### 3.4.2 Algoritmo Determinístico

O decodificador é baseado em um algoritmo determinístico que coloca as peças, uma a uma, na placa. O gerenciamento dos retângulos livres para colocação das peças é baseado em uma lista de retângulos livres, como descrito por Lai e Chan (1997) na seção 3.2. Um retângulo

livre é adicionado à lista de retângulos livres se não estiver contido em nenhum outro retângulo livre da placa. Cada vez que uma peça é colocada em um retângulo livre, novos retângulos livres são gerados.

As seguintes fases são aplicadas para decodificar a proposta de solução do problema:

(1) Decodificação da sequência de corte: essa primeira fase decodifica parte da proposta de solução do problema chamada de BPS, que é a sequência na qual as peças serão cortadas na placa.

(2) Decodificação das heurísticas de colocação: a segunda fase decodifica parte da proposta de solução do problema chamada de VPH, definindo qual heurística de colocação será adotada para cada peça a ser cortada na placa.

(3) Decodificação da orientação das peças: a terceira fase decodifica parte da proposta de solução chamada de VBO, definindo como deve ser a orientação de cada peça ao ser cortada na placa.

(4) Formação do padrão de corte: a quarta fase faz uso do BPS, VPH e VBO, decodificados na fase 1, 2 e 3, respectivamente. Nessa fase, constrói-se o padrão de corte das peças na placa, correspondente à proposta de solução antes da decodificação.

Segundo Gonçalves e Resende (2013), a decodificação das primeiras variáveis  $n$  de cada proposta de solução, em uma sequência de corte de peças (*BPS*), é realizada pela classificação, em ordem crescente, dos valores dessas variáveis. Na Figura 22, ilustra-se um exemplo do processo de decodificação para o *BPS*. Neste exemplo, há 8 peças. A classificação das variáveis com seus valores em ordem crescente produz o seguinte *BPS* = (5, 8, 3, 1, 4, 2, 6, 7).

**Figura 22** - Decodificação da Sequência de Peças Cortadas

Peças a serem cortadas	1	2	3	4	5	6	7	8
Valor gerado aleatoriamente para as variáveis que representam as peças a serem cortadas	0.45	0.67	0.35	0.49	0.07	0.78	0.87	0.17
Decodificar (Classificar em ordem crescente) as variáveis que representam as peças a serem cortadas	0.07	0.17	0.35	0.45	0.49	0.67	0.78	0.87
BPS (Sequência de peças cortadas)	5	8	3	1	4	2	6	7

Fonte: Adaptado de Gonçalves e Resende (2013) pela autora.

As variáveis que constam na segunda fase da decodificação são as de  $n + 1$  até  $2n$ , que representam a colocação das peças na placa.

Os testes realizados pelos autores Gonçalves e Resende (2013) consideravam o problema de corte bidimensional e tridimensional. Quando o problema considera três dimensões, faz todo sentido que se utilize a decodificação da colocação das peças em um *container*, onde se pode decodificar a colocação, utilizando-se a heurística *Back-Bottom-Left* (BBL) ou a heurística *Back-Left-Bottom* (BLB). Em um problema que utiliza duas dimensões, não faz sentido a escolha da heurística, pois subtraindo-se uma dimensão dele, a heurística é a mesma. Contudo, esse passo é de grande importância para o entendimento do algoritmo decodificador como um todo. Ainda que neste trabalho o problema estudado seja bidimensional, a descrição do processo de decodificação da colocação das peças no *container* será detalhada.

Durante a tentativa de colocar uma peça em um *container*, Gonçalves e Resende (2013) utilizaram uma lista *RL* de maiores espaços vazios (*Empty Maximal-Spaces* - EMSs), ou seja, maiores retângulos livres para preenchimento com peças. Maiores retângulos são representados pelos seus vértices, com mínimas e máximas coordenadas ( $\{x_i, y_i, z_i\}$  e  $\{X_i, Y_i, Z_i\}$ ), respectivamente. Ao procurar um lugar para colocar uma peça, Gonçalves e Resende (2013) consideraram apenas as coordenadas correspondentes aos vértices EMSs com coordenadas mínimas ( $x_i, y_i, z_i$ ). Para gerar e manter o controle dos EMSs, os autores fizeram uso do processo de diferença proposto por Lai e Chan (1997). A descrição detalhada desse processo pode ser encontrada na seção 3.2 deste trabalho.

Segundo Gonçalves e Resende (2013), depois que a lista *RL* foi atualizada pelo processo de diferença, outro processo proposto por Lai e Chan (1997) precisou ser aplicado: o processo de eliminação, em que os EMSs, com uma única dimensão ou aqueles que estão totalmente inscritos por outros EMSs, devem ser eliminados da lista *RL*. Esse processo é mais demorado no algoritmo de corte. Para reduzir o tempo de computação para essa tarefa, Gonçalves e Resende (2013) adicionaram as seguintes regras para o processo de diferença:

- Se a menor dimensão de um EMS recém-criado for menor do que aquela de cada uma das peças restantes para serem cortadas, não adicione à lista *RL*. Nota-se que os EMSs com uma única dimensão serão automaticamente removidos por esse passo.
- Se o volume de um EMS recém-criado for menor do que o volume de cada uma das peças restantes para serem alocadas, não adicione o EMS à lista *RL*. Nota-se que essa regra será muito importante para a eliminação dos EMSs quando as peças não se encaixarem e não forem removidas pela regra anterior.

Inicialmente, segundo Gonçalves e Resende (2013), foram considerados apenas os procedimentos BBL que ordenam os EMSs de uma placa, de tal forma que  $EMS_i < EMS_j$  se  $x_i < x_j$ , ou se  $x_i = x_j$  e  $z_i < z_j$ , ou se  $x_i = x_j$ ,  $z_i = z_j$  e  $y_i < y_j$ . Em seguida, escolhe-se o primeiro EMS no qual a peça a ser cortada fique totalmente inserida, utilizando-se qualquer uma das possíveis orientações. No entanto, como observado por Liu e Teng (1999), Gonçalves e Resende (2013) verificaram que algumas soluções ótimas não podiam ser construídas pela heurística de colocação BBL. Para superar esse ponto fraco, os autores combinaram a heurística de colocação BBL com a BLB. Essa combinação de heurísticas ordena os EMSs de um *container*, de tal forma que  $EMS_i < EMS_j$  se  $x_i < x_j$  ou se  $x_i = x_j$  e  $y_i < y_j$ , ou se  $x_i = x_j$ ,  $y_i = y_j$  e  $z_i < z_j$ .

Em seguida, escolhe-se o primeiro EMS no qual a peça a ser cortada fica totalmente inserida, utilizando-se qualquer uma das orientações possíveis. Em suma, a estratégia de posicionamento de Gonçalves e Resende (2013) usa duas heurísticas de colocação, a *Back-Bottom-Left* ou a *Back-Left-Bottom* para construir a sequência de alocação de peças em um *container*.

A decodificação do vetor de heurísticas de colocação (*VPH*) é realizado por  $i = 1, \dots, n$ , usando-se a expressão:

$$VPH_i = \begin{cases} BBL & \text{se } h_{n+i} \leq \frac{1}{2}, \\ BLB & \text{se } h_{n+i} > \frac{1}{2}. \end{cases}$$

O *VPH* é usado pelo processo de colocação para selecionar qual heurística de colocação deverá ser aplicada em cada peça.

A terceira fase de decodificação é aplicada ao vetor de orientação das peças (*VBO*). A decodificação é obtida por  $VBO_i = o_{2n+i}$ , considerando  $i = 1, \dots, n$ , onde se for encontrado, em cada posição do vetor *VBO*, um valor menor ou igual a 0,5, significa que a peça não sofrerá rotação em sua orientação. Entretanto, nos casos em que o valor encontrado for maior que 0,5, a peça sofrerá rotação, logo a dimensão do comprimento passará a ser considerada altura, e a dimensão da altura passará a ser considerada comprimento. Nessa fase, a decodificação, na verdade, é a transcrição dos valores das variáveis  $o_{2n+i}$  até  $o_{3n}$ , os quais constam na proposta de solução codificada para o vetor *VBO*.

A última fase no processo de decodificação é a formação do padrão de corte. Essa fase segue um processo sequencial, no qual corta uma peça na placa em cada estágio. A ordem em que as peças são cortadas é definida pelo BPS. A colocação da peça, dentro do *container*, é

definida pelo VPH, e o *VBO* é usado pelo processo de formação do padrão de corte para determinar qual orientação será aplicada em cada peça.

O processo combina os seguintes elementos: os vetores BPS, VPH, e VBO, a lista *RL* de maiores retângulos livres e as heurísticas de colocação BBL e BLB. Cada estágio é composto pelos seis passos seguintes:

1. Seleção da peça;
2. Seleção da heurística de colocação;
3. Seleção do maior retângulo livre da placa;
4. Seleção da orientação da peça;
5. Corte da peça;
6. Atualização da informação do estado.

Ao final, tem-se o padrão de corte que é a solução real do problema, onde se deve obter sempre a mesma configuração de corte, independentemente se a proposta de solução foi interpretada por uma pessoa ou por uma máquina. Porém, no caso da proposta de solução decodificada encontrada por Gonçalves e Resende (2013), para que uma pessoa ou computador consiga chegar à solução real do problema em estudo, não basta que se tenha as três parcelas do vetor de proposta de solução decodificadas, pois sem o conhecimento das regras do algoritmo determinístico não se consegue reproduzir o mesmo padrão de corte encontrado como solução final. Logo, por mais que se tenha as três parcelas do vetor de solução decodificadas, é preciso que a máquina ou o homem execute, em conjunto, o algoritmo determinístico decodificador, isso significa que sempre existirá essa dependência em relação ao decodificador.

### **3.4.3 Definição de Recombinação de Soluções**

O processo de recombinação proposto por Gonçalves e Resende (2013) é baseado em chaves aleatórias viciadas. A partir de duas soluções geradoras, obtém-se a base para encontrar um descendente: a primeira solução geradora deve pertencer, necessariamente, ao grupo de elite de soluções do problema, por esse motivo o processo tem, em parte do seu nome, a palavra viciada; a outra solução geradora pode ser ou não do grupo de elite.

Depois de escolhidas as soluções geradoras, elas serão recombinadas, criando-se somente um descendente. Esse processo se assemelha ao de aplicação de estrutura de vizinhança, onde é encontrado o vizinho de uma proposta de solução. No processo de recombinação, obtém-se uma solução semelhante aos pais, mas chamada, aqui, de descendente. Segundo Gonçalves e Resende (2013), supondo-se que foi selecionada a solução de elite A e a



solução B para participarem da recombinação, adicionalmente escolhe-se o parâmetro  $q_e$ , que indica a probabilidade de que uma variável da solução de elite A seja escolhida para ser incorporada ao descendente V que está sendo construído. Assim, para gerar as  $n$  variáveis do descendente V, deve-se decidir se copia para ele o valor armazenado em A ou em B. Sendo  $p_i$  um valor entre 0 e 1, gerado aleatoriamente. Nesse caso, entende-se que  $i = 1, \dots, n$ , se  $p_i \leq q_e$ , então copia-se o valor armazenado na solução A para V. Caso contrário, copia-se o valor armazenado em B para V. Desse processo, origina-se a outra parte do nome de tal recombinação que é de chaves aleatórias. Segundo Gonçalves e Resende (2013), o valor usado para  $q_e$  é igual a 0,7. Portanto, em cada caso, existe a probabilidade de 0,7 para que o valor armazenado em A seja escolhido e copiado no descendente V que está sendo gerado. Obviamente, essa estratégia permite que a maioria das variáveis copiadas em V sejam armazenadas na solução de elite. Na Tabela 3, ilustra-se a recombinação que dá origem ao descendente.

**Tabela 3** - Recombinação de Soluções

Solução de elite – A	0.36	0.45	0.12	0.64	0.91	0.84
Solução - B	0.18	0.31	0.51	0.21	0.74	0.29
$p = N^\circ$ aleatório no intervalo [0,1]	0.39	0.87	0.17	0.96	0.55	0.63
Comparação com $q_e = 0,7$	<	>	<	>	<	<
Solução descendente V	0.36	0.31	0.12	0.21	0.91	0.84

Fonte: Adaptado de Gonçalves e Resende (2013) pela autora.

### 3.5 REPRESENTAÇÃO PROPOSTA PELA AUTORA

Nesta seção, apresenta-se uma proposta criada pela autora deste trabalho, para representar soluções do problema de corte bidimensional não guilhotinado com restrições.

#### 3.5.1 Representação da Proposta de Solução

Analisando-se a representação da proposta de solução apresentada por Lai e Chan (1997), é possível verificar que, após a solução codificada passar pelo algoritmo de decodificação, tem-se um conjunto de informações que dão origem ao padrão de corte das peças na placa. Após o corte das peças que constam no vetor ordenado de corte, tem-se um conjunto de informações composto por: vetor ordenado de corte, onde são identificadas quais as peças que realmente serão cortadas na placa e sua ordem de corte; vetor com os retângulos livres,

produzidos pelas peças cortadas na placa; valor objetivo ou *fitness* do desperdício de área da placa.

O segundo vetor (retângulos livres produzidos pelas peças cortadas), que compõe as informações sobre o padrão de corte, é muito dinâmico e, o tempo todo, sofre modificações que exigem muito espaço de armazenamento e processamento, fazendo com que o algoritmo se torne lento, ao atualizar a lista de retângulos livres, todas as vezes que uma peça é cortada na placa.

O vetor traz consigo, na etapa corrente, os retângulos livres da etapa anterior, atualizando somente os espaços que foram ocupados pela nova peça cortada e acrescentando novos retângulos livres, ou seja, existe uma redundância de informações que é sustentada a cada peça cortada. Exemplo: se com o corte da primeira peça na placa forem gerados os retângulos livres  $R_1$  e  $R_2$ , após cortar a segunda peça no retângulo livre  $R_2$  têm-se os seguintes retângulos livres:  $R_1$ ,  $R_3$  e  $R_4$ . Ao cortar a terceira peça no retângulo livre  $R_3$ , a lista de retângulos livres gerada por esta peça será  $R_1$ ,  $R_4$ ,  $R_5$ ,  $R_6$  e assim sucessivamente.

Além da redundância de informações armazenadas, existe um segundo problema. Se uma pessoa ou uma máquina de uma indústria de corte de matéria-prima executar o padrão de corte com as informações propostas por Lai e Chan (1997) para representar a solução do problema de corte bidimensional não guilhotinado, essa não será uma tarefa trivial, mesmo o padrão de corte fornecendo todas as informações necessárias para a conclusão de tal tarefa. Isso se dá pelo fato das informações não serem precisas e, por isso, necessitem de uma análise da disposição dos retângulos livres resultantes a cada corte de uma peça na placa.

Com relação à proposta de Alvarez-Valdes, Parreño e Tamarit (2007), é possível encontrar, ao final da execução do algoritmo decodificador, uma proposta de solução do problema de corte bidimensional não guilhotinado composta por quatro listas de informações: uma com a sequência de blocos de peças cortadas na placa; uma com a quantidade de peças que formam cada bloco; uma de retângulos livres restantes na placa, produzidos após o corte de todos os blocos; uma com o valor total das peças cortadas na placa.

Verifica-se que os autores não armazenam a lista de retângulos livres gerados por cada peça cortada na placa, a informação que compõe a representação da proposta de solução do problema é uma lista de retângulos livres que sobraram ao final do processo de corte de todas as peças. Ou seja, após cortar a última peça, verifica-se quais os retângulos livres que sobraram, incluindo-os em uma lista de retângulos livres que compõem o padrão de corte, porém, se o conjunto de informações que compõem a proposta de solução do problema fosse passado para uma terceira pessoa ou a uma máquina que não tivesse o conhecimento do algoritmo

decodificador proposto por Alvarez-Valdes, Parreño e Tamarit (2007), tampouco de suas regras e particularidades, certamente o corte das peças não aconteceria na mesma localização dentro da placa, conforme a solução dada pelo algoritmo decodificador. Nesse sentido, percebe-se que há uma dependência das regras e especificidades do algoritmo para a tomada de decisão sobre qual retângulo livre escolher em cada etapa do processo de corte de peças na placa. Assim, sem essa informação, muda-se a configuração do padrão de corte e, certamente, não será produzido o mesmo padrão ao final do processo. Considerando-se a situação em que uma terceira pessoa ou máquina conhecesse o algoritmo decodificador e tivesse acesso ao padrão de corte dado por ele como solução final para o problema em estudo, essa terceira entidade, ao interpretar as informações dadas pelo padrão de corte, teria, necessariamente, que executar o algoritmo decodificador em conjunto, sempre, em todas as situações de interpretação do padrão, gerando um tempo de processamento maior para a interpretação do padrão.

A mesma situação ocorre ao se analisar o padrão de corte dado como solução para o problema de corte bidimensional não guilhotinado, segundo Gonçalves e Resende (2013), pois, no caso da representação da proposta de solução dada por esses autores, ela é composta por um vetor com três parcelas. A primeira delas representa a sequência de corte das peças na placa; a segunda, a heurística de colocação adotada para cada peça na placa; a terceira, a orientação de cada peça que foi cortada na placa. Unindo-se as informações encontradas nas três parcelas desse vetor, tem-se um padrão de corte das peças, na placa, como solução para o problema original. A situação de indefinição, ao interpretar o padrão de corte dado pelo algoritmo decodificador de Gonçalves e Resende (2013), repete-se, assim como aconteceu com as propostas de representação de solução do problema dada pelos outros autores analisados anteriormente. Para que o padrão de corte possa ser interpretado por terceiros de maneira que gere, ao final, o mesmo resultado encontrado pelo algoritmo decodificador, esses terceiros teriam que deter o conhecimento sobre as regras e as particularidades do algoritmo decodificador.

A autora deste estudo apresenta uma proposta de representação da solução do problema de corte bidimensional não guilhotinado que se baseia na proposta de Alvarez-Valdes, Parreño e Tamarit (2007). Todavia, caracteriza-se como uma representação mais realista que, se interpretada por uma pessoa de fora do processo, de fato possibilitará a reprodução do padrão de corte dado como solução pelo algoritmo decodificador.

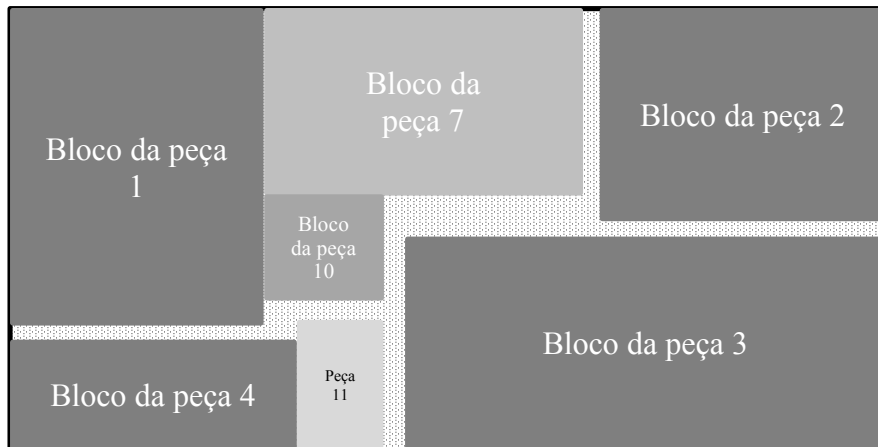
A proposta da autora para a representação da solução difere-se da proposta de Lai e Chan (1997) e da proposta de Gonçalves e Resende (2013), por ser de simples interpretação. As informações que compõem essa proposta podem ser interpretadas até mesmo por pessoas

com nenhuma experiência relacionada a esse problema. Outro ponto importante nela é o fato de não haver necessidade de armazenar dados replicados, gerando um volume menor de processamento e de armazenamento.

Se considerar que uma máquina faria o trabalho de cortar uma placa segundo o padrão de corte proposto pela autora, essa máquina não necessitaria executar o decodificador após ter conhecimento do padrão de corte, pois o padrão, proposto neste trabalho, depois de conhecido, torna-se autossuficiente. Por outro lado, as propostas dos outros autores analisados necessitam do conhecimento das regras do decodificador para que o padrão de corte seja interpretado corretamente. Sendo assim, as propostas dos autores estudados requerem que o computador refaça o processamento do decodificador, ainda que tenham o padrão de corte conhecido, pois as informações nele contidas são insuficientes para reproduzir o mesmo padrão.

A proposta da autora consiste em fornecer um padrão de corte como representação da solução do problema, sendo ele composto por quatro listas ou vetores: a primeira delas contendo a sequência das peças cortadas; a segunda obedecendo à ordem das peças da primeira, apresentando a quantidade de cópias de cada peça cortada na placa e dando origem aos blocos; a terceira contendo a orientação que as peças de cada bloco assumem ao serem cortadas como bloco na placa; a quarta contendo as coordenadas dos blocos dentro da placa, levando-se em consideração que a origem da placa encontra-se em seu canto inferior esquerdo. Portanto, o padrão de corte representado pela proposta de solução dada pela autora deste trabalho, se entregue a uma terceira pessoa ou a uma máquina, permitirá a reprodução fiel da execução dos cortes que o algoritmo decodificador processou, porém sem a necessidade de se conhecer o algoritmo. Adicionalmente às quatro listas que compõem a representação proposta pela autora para solucionar o problema, pode ser incorporado o valor da função objetivo ou *fitness*, o qual representa a soma dos valores das peças cortadas na placa.

Na Figura 23, ilustra-se a diferença entre as representações de proposta de solução para o problema de corte bidimensional não guilhotinado feitas por Lai e Chan (1997) e por Alvarez-Valdes, Parreño e Tamarit (2007) e a representação de proposta de solução feita pela autora do trabalho, na qual as peças foram cortadas em uma placa com dimensão (120 x 100). A origem da placa encontra-se no canto inferior esquerdo, as coordenadas na representação da solução proposta pela autora trazem quatro números. Os dois primeiros referem-se ao canto inferior esquerdo do bloco, cortado na placa, e os dois últimos representam o canto superior direito do bloco.

**Figura 23** - Diferença Entre as Três Representações de Solução para o Problema

<b>Representação de Solução Segundo Lai e Chan (1997)</b>	<b>Representação de Solução Segundo Alvarez-Valdes, Parreño e Tamarit (2007)</b>	<b>Representação de Solução Proposta pela autora deste trabalho</b>
Lista ordenada de corte das peças: 04, 04, 04, 02, 02, 02, 02, 01, 01, 01, 01, 01, 01, 03, 03, 03, 03, 07, 07, 10, 11	Lista ordenada de corte das peças: 04, 02, 01, 03, 07, 10, 11	Lista ordenada de corte das peças: 04, 02, 01, 03, 07, 10, 11
Valor Objetivo do Desperdício da placa: 6 unidades de área da placa	Qtd. de peças que formam os blocos: 03, 04, 06, 04, 02, 01, 01	Qtd. de peças que formam os blocos: 03, 04, 06, 04, 02, 01, 01
Lista de retângulos livres: Relativo à primeira peça 04 ( $R_1, R_2$ ) Relativo à segunda peça 04 ( $R_2, R_3$ ) Relativo à terceira peça 04 ( $R_2, R_4$ ) Relativo à primeira peça 02 ( $R_5, R_6, R_7$ ) Relativo à segunda peça 02 ( $R_8, R_9, R_{10}$ ) Relativo à terceira peça 02 ( $R_{11}, R_{12}, R_{13}$ ) Relativo à quarta peça 02 ( $R_{14}, R_{15}, R_{16}$ ) Relativo à primeira peça 01 ( $R_{14}, R_{16}, R_{17}, R_{18}$ ) Relativo à segunda peça 01 ( $R_{14}, R_{16}, R_{17}, R_{19}$ ) Relativo à terceira peça 01 ( $R_{14}, R_{16}, R_{17}, R_{20}$ ) Relativo à quarta peça 01 ( $R_{14}, R_{16}, R_{17}, R_{21}$ ) Relativo à quinta peça 01 ( $R_{14}, R_{16}, R_{17}, R_{22}$ ) Relativo à sexta peça 01 ( $R_{14}, R_{16}, R_{17}, R_{23}$ ) ...	Soma dos valores das peças: 194	Orientação dos blocos: 0, 0, 0, 1, 0, 0, 1 (o número 0 (zero) significa que não houve rotação, o número 1 (um) significa que houve rotação)
	Lista de retângulos livres: 03 x 55 38 x 05 16 x 05 05 x 01 30 x 10 03 x 45 38 x 05	Soma dos valores das peças: 194  Coordenadas onde os blocos devem ser cortados na placa: Bloco 4:((000, 000), (038, 023)) Bloco 2:((082, 050), (120, 100)) Bloco 1:((000, 028), (033, 100)) Bloco 3:((052, 000), (120, 045)) Bloco 7:((033, 055), (079, 100)) Bloco 10:((033, 034), (049, 055)) Bloco 11:((038, 000),

		(049, 029))
--	--	-------------

Fonte: Elaborado pela autora.

Nos próximos capítulos, serão apresentados todos os itens propostos pela autora deste trabalho, os algoritmos meta-heurísticos utilizados para solucionar o problema de corte bidimensional não guilhotinado, o processo de decodificação de proposta de solução e a descrição sobre como ocorre a recombinação de soluções para encontrar configurações descendentes ou vizinhas.

## 4 ALGORITMO GENÉTICO ESPECIALIZADO PARA O PROBLEMA DE CORTE BIDIMENSIONAL NÃO GUILHOTINADO

Neste capítulo, o leitor encontrará o algoritmo genético sugerido pela autora do trabalho, com vistas à resolução do problema de corte bidimensional não guilhotinado. Essa proposta tem como base o algoritmo genético de chaves aleatórias viciadas, indicado por Gonçalves e Resende (2013), acrescido de melhorias realizadas pela autora como, por exemplo, a formação de blocos e o escaneamento de retângulos livres.

O capítulo inicia com uma introdução sobre o algoritmo genético. Na seção 4.2, mostra-se o funcionamento do algoritmo genético tradicional. Na seção 4.3, é tratada a questão do algoritmo genético de chaves aleatórias viciadas proposto por Gonçalves e Resende (2013). Por fim, na seção 4.4, é apresentado o algoritmo genético de chaves aleatórias viciadas especializado. Esse algoritmo especializado é uma das propostas meta-heurísticas deste trabalho para a resolução do problema de corte bidimensional não guilhotinado.

### 4.1 INTRODUÇÃO

Conforme mencionado nos capítulos anteriores, as técnicas meta-heurísticas, em especial os algoritmos genéticos, são amplamente aplicadas na resolução do problema de corte bidimensional não guilhotinado e nos problemas de otimização em geral. Como o problema de corte bidimensional é considerado *NP-hard*, é perfeitamente aceitável o emprego de heurísticas e de meta-heurísticas para a sua resolução. Os algoritmos genéticos normalmente associam à sua estrutura tradicional um procedimento complementar para a resolução do problema aqui estudado.

O algoritmo genético, segundo Soares (1997), é uma técnica de busca que visa à obtenção de um resultado ótimo, ou de boa qualidade, por meio de propostas de solução para um determinado problema. Inicialmente, foi formulado por John Holland, nos anos 60, sendo posteriormente desenvolvido na Universidade de Michigan, por ele e seus alunos, em meados dos anos 70. Holland (1975) tinha como objetivo principal dedicar-se ao estudo informal do fenômeno da evolução, como ocorre na natureza, e desenvolver maneiras para incorporá-los aos sistemas computacionais.

A evolução das espécies é determinada por um processo de seleção que leva à sobrevivência dos indivíduos que são geneticamente melhor adaptados para superar os problemas existentes no meio ambiente em que vivem. Quando uma determinada população, por exemplo, necessita adaptar-se às novas condições impostas por esse ambiente, sejam elas

quais forem (a falta de comida, a questão espacial ou qualquer outro recurso essencial), os indivíduos que mais se adaptarem a essas novas condições sobreviverão, enquanto os mais fracos serão excluídos e extintos daquele meio.

Isso acontece porque os seres mais fortes possuem, de forma acentuada, algumas das características imprescindíveis à sobrevivência, principalmente quando comparados aos mais fracos. Por herança, essas características acentuadas provavelmente serão passadas para seus descendentes e, assim, eles terão grandes chances de sobreviverem também. Por outro lado, indivíduos fortes podem surgir da exploração de alguma característica ainda não desenvolvida na população. Se a natureza tentasse descobrir essas novas características, por meio da seleção dos mais aptos e da recombinação dentro de um mesmo grupo, certamente não teria sucesso, visto que, depois de muitas gerações, todos os membros de uma comunidade compartilhariam praticamente do mesmo código genético. Para contornar o problema, a natureza insere material genético diferente nesses membros, por meio do processo conhecido como mutação. Se o ser que sofreu mutação estiver tão capacitado à sobrevivência quanto os atuais, suas chances são grandes em um futuro processo de seleção.

Com essa motivação, os algoritmos genéticos nasceram seguindo uma metodologia baseada nos mecanismos da seleção natural e da evolução, mecanismos esses que foram propostos na teoria de Darwin (1859).

A evolução ocorre quando novos indivíduos são introduzidos na população. Com ela, pretende-se, em cada geração, encontrar soluções mais qualificadas. Para isso, o algoritmo genético realiza, segundo Soares (1997), o chamado ciclo geracional, ou seja, instauram-se os operadores de seleção, a recombinação e a mutação. Na seleção, os melhores indivíduos são privilegiados com maior probabilidade de participar da nova população; na recombinação, é realizada a troca de material genético entre as configurações; na mutação, novos genes são incorporados à população.

Portanto, para que a população possa evoluir, é necessário definir uma representação adequada dos seus indivíduos. Esses indivíduos constituem as configurações e, no processo de otimização, as configurações representam as soluções candidatas dos problemas a serem resolvidos. Assim, a escolha da codificação está diretamente relacionada às características do problema.

Os algoritmos genéticos encontram, de maneira adequada, os ótimos locais presentes nos problemas que possuem elevada quantidade de alternativas, pois, por meio da população considerada um conjunto de configurações, é possível avaliar, simultaneamente, várias regiões pertencentes ao mesmo espaço de busca. Essa característica, entre outras, torna-se responsável



pelo bom desempenho da meta-heurística algoritmo genético em resolver problemas de otimização.

## 4.2 FUNCIONAMENTO DO ALGORITMO GENÉTICO TRADICIONAL

Antes de iniciar uma explicação mais detalhada acerca do funcionamento do algoritmo genético tradicional, é necessário o conhecimento dos conceitos de alguns termos comuns empregados quando se utiliza algoritmo genético. Tais termos técnicos serão definidos a seguir, com o intuito de promover um melhor entendimento do trabalho realizado.

### 4.2.1 Sistema Natural x Algoritmo Genético

Para melhor entendimento dos termos usados na genética natural e aplicados no algoritmo genético, será realizada aqui uma analogia entre os termos. Como o problema analisado neste trabalho é o corte bidimensional não guilhotinado, os termos conceituados serão direcionados a esse contexto.

Segundo Vendramini (2007) um indivíduo ou cromossomo corresponde a uma proposta de solução codificada para o problema em estudo. Essa proposta caracteriza-se como um vetor composto pela ordem das peças cortadas na placa e pela orientação de cada uma delas. Cada gene do cromossomo corresponde a uma peça cortada na placa, e a população ou geração compreende o conjunto de cromossomos. No algoritmo genético, o processo de obtenção de uma nova população de cromossomos também pode ser descrito como formação de uma nova geração de cromossomos. Já os termos recombinação, mutação e população estão diretamente ligados aos indivíduos.

Na Tabela 4, mostra-se a relação existente entre as entidades de sistemas naturais e os termos usados no algoritmo genético.

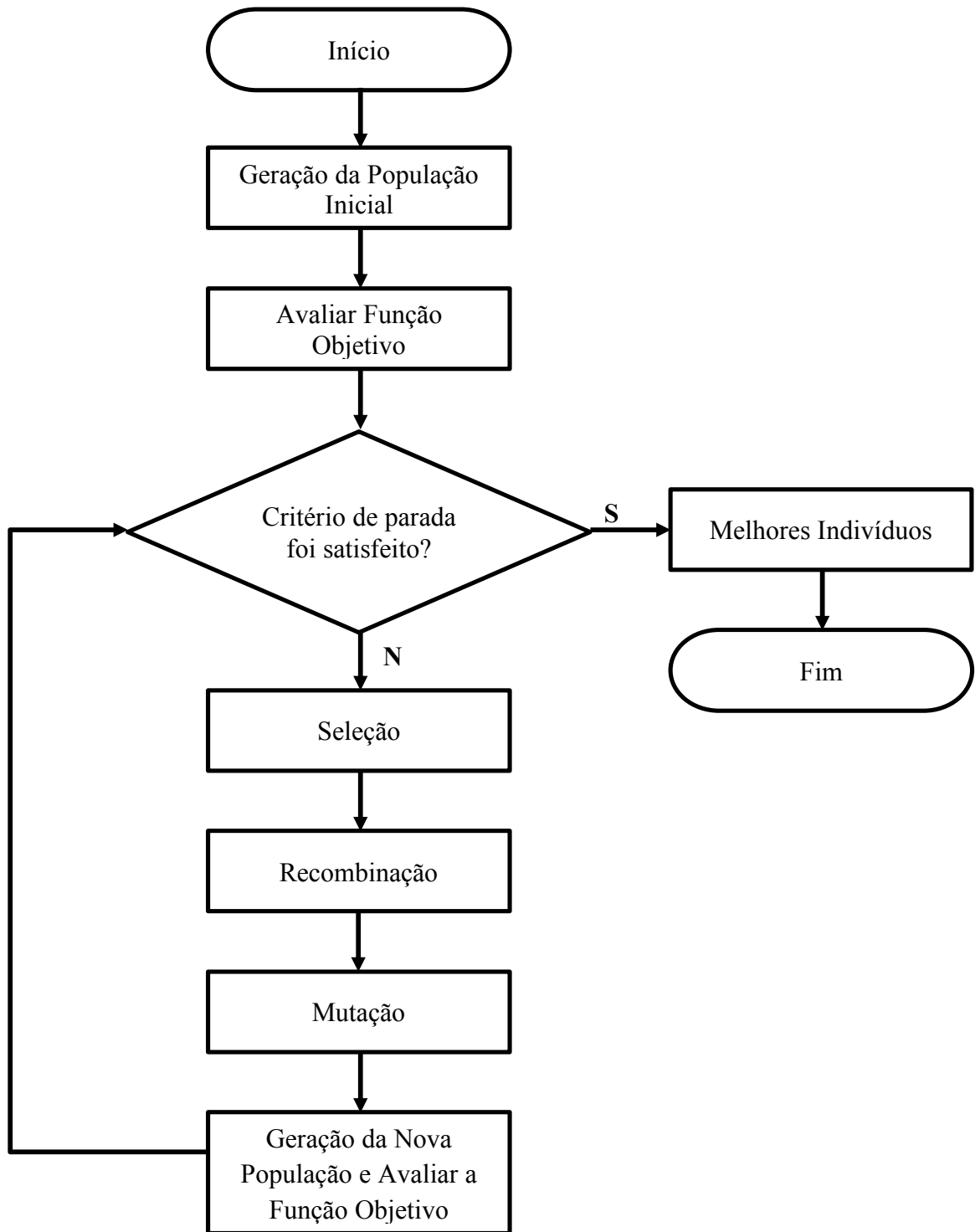
**Tabela 4** - Analogia Entre Sistemas Naturais e os Algoritmos Genéticos

Genética Natural	Algoritmo genético
Gene	Peça
Alelo	Índice de cada peça
Indivíduo (Cromossomo)	Configuração ou proposta de solução codificada para o problema.
População	Conjunto de propostas de solução codificadas do problema.
Genótipo	Estrutura, cromossomo
Fenótipo	Estrutura decodificada

Fonte: Elaborado pela autora.

Segundo Vendramini (2007), o algoritmo, na sua forma mais simples, deve cumprir as seguintes etapas:

1. Representar adequadamente uma configuração do problema. A mais popular é a representação em codificação binária, em que os operadores genéticos de recombinação e mutação são facilmente simulados;
2. Encontrar uma forma correta de avaliar a função objetivo ou o seu equivalente (*fitness*) e de identificar as configurações de melhor qualidade;
3. Desenvolver uma estratégia de seleção das configurações que atribua às configurações de melhor qualidade uma maior participação na formação das configurações da nova população (nova geração);
4. Criar um mecanismo que permita implementar o operador genético de recombinação;
5. Implementar o operador genético de mutação e terminar de gerar a nova população;
6. Parar quando o critério de parada for atendido. Caso contrário, voltar ao passo 2. A seguir, a Figura 24 mostra o fluxograma do funcionamento do algoritmo genético tradicional.

**Figura 24** - Fluxograma do Funcionamento dos Algoritmos Genéticos Tradicionais

Fonte: Elaborado pela autora.

## **4.2.2 Funcionamento do Algoritmo Genético**

No decorrer desta seção, encontram-se explicados, com mais detalhes, os principais tópicos do algoritmo genético, a partir da codificação e representação das configurações.

### **4.2.2.1 Codificação**

Segundo Vendramini (2007), para trabalhar com um algoritmo genético, é preciso representar as soluções candidatas por meio de uma codificação. A forma de implementar a codificação depende do problema a ser estudado, da natureza das variáveis de decisão do problema ou da forma de representar uma configuração usada no problema. Existem problemas que trabalham com variáveis inteiras, reais ou, ainda, binárias.

O que deve ser realmente levado em consideração quando se codifica um problema, a fim de ser resolvido por algoritmos genéticos, são as suas características. Essas características determinam a melhor maneira de codificar as soluções candidatas, se é melhor codificar binariamente, usar variáveis inteiras ou variáveis reais.

### **4.2.2.2 Função Objetivo**

O algoritmo genético, antes de ser desenvolvido, exige a análise e o estudo do problema a ser otimizado. Faz-se necessário um estudo minucioso sobre os fatores do problema, as variáveis envolvidas, assim como todas as informações que influenciam direta e indiretamente nesse problema.

Segundo Vendramini (2007), a função objetivo retorna o desempenho de cada configuração avaliada ou a aptidão de cada uma, ou seja, ela é utilizada a fim de verificar a qualidade de cada configuração. Esse processo é necessário para que se possa, mais tarde, comparar e verificar quais são as melhores configurações, aplicando a elas o operador de seleção.

Muitas vezes, quando se precisa padronizar os resultados da função objetivo, para que posteriormente eles sejam analisados e submetidos ao processo de seleção, encontram-se algumas dificuldades em representar a função objetivo no seu formato original. Segundo Soares (1997), quando o objetivo da função não é maximizar e, sim, minimizar, tem-se um exemplo claro dessa dificuldade, haja vista que para a maioria dos operadores de seleção precisa-se trabalhar com valores padronizados, com exceção do processo de seleção por torneio, que não

necessita padronizar valores.

Portanto, os algoritmos genéticos trabalham em termos de maximização e, muitas vezes, o objetivo é minimizar uma função. Em casos de minimização, é necessário transformar a função objetivo em uma outra função, chamada de função adaptativa ou *fitness function*, em que esse problema seja contornado e possa, de alguma maneira, comparar resultados e aplicar o operador de seleção. Entretanto, como já foi ressaltado, para determinados tipos de seleção, como a seleção por torneio, não existe problema em trabalhar com a minimização. Portanto, trabalha-se sem dificuldades com problemas de maximização ou minimização, sem necessidade de transformação ou de padronização.

A função objetivo também está diretamente ligada à infactibilidade de uma configuração. Ao avaliar uma configuração, o resultado pode ser factível ou infactível, e essa informação deve ser incorporada à função objetivo. Para corrigir esse problema, pode-se usar uma penalidade para as infactibilidades encontradas ou, ainda, aplicar os operadores genéticos prevendo esse tipo de solução infactível (VENDRAMINI, 2007).

#### **4.2.2.3 Métodos de Seleção**

Segundo Vendramini (2007), o operador de seleção é responsável pela escolha das configurações que serão submetidas aos operadores genéticos, como a recombinação e a mutação, e as configurações resultantes dessas operações farão parte da nova geração. Não é um bom procedimento favorecer sempre a seleção do melhor, muito menos uma escolha aleatória. Por um lado, há a possibilidade de ocorrer convergência prematura; por outro, a pesquisa é aleatória, deixando de explorar as informações contidas na população. A seguir, tem-se a descrição de alguns métodos de seleção.

**Seleção Proporcional:** o método da seleção proporcional é um método de seleção simples, mas que gera alguns problemas na seleção de configurações, que podem ter uma participação na próxima geração por meio de seus descendentes. As configurações de uma geração são selecionadas para a próxima geração utilizando uma roleta.

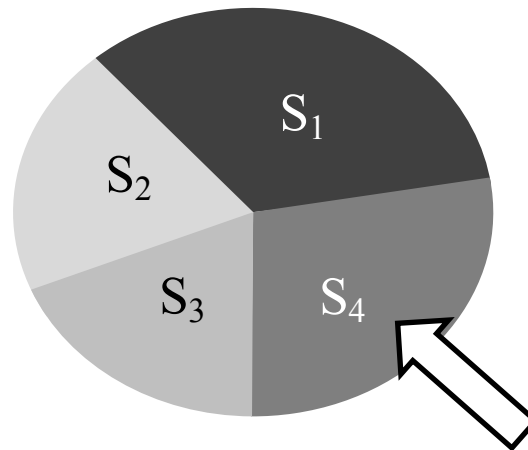
Cada configuração da população é representada, na roleta, por uma fatia proporcional à razão entre o valor da função objetivo da configuração, dividido pela média da função objetivo da população. A roleta é acionada determinado número de vezes, esse método é definido pelo tamanho da população. A cada giro da roleta, uma configuração é apontada pela agulha e, então, selecionada.

Existem outros métodos de seleção que utilizam a roleta e são derivados do método de

seleção proporcional, um exemplo é a **Seleção Estocástica de Resíduos**. Na Figura 25, é exibida a representação da roleta usada nesse método de seleção.

**Seleção por Torneio:** esse método também é muito simples e tem a vantagem de eliminar os problemas que são encontrados usando a seleção proporcional. Ele retorna a melhor configuração, ou seja, aquela que traz uma função objetivo com valor melhor entre  $j$  indivíduos participantes. Esse método busca dificultar, porém sem eliminar, as possibilidades de uma configuração com baixo desempenho de ser escolhida. A seleção por torneio é a preferida pelos pesquisadores, porque apresenta bom desempenho e facilidade de implementação computacional. Existem, ainda, outros métodos de seleção, tais como: a **Seleção Determinística** e a **Baseada em Ordenamento**.

**Figura 25** - Modelo da Roleta do Método de Seleção Proporcional



Fonte: Elaborado pela autora.

#### **4.2.2.4 Métodos de Recombinação**

Segundo Vendramini (2007), a recombinação é o principal operador genético que atua sobre as configurações selecionadas. Segundo Soares (1997), ela é a maior responsável pela troca de material genético entre duas configurações participantes, fazendo que novas e melhores gerações apareçam. Por esse motivo, fica claro que a recombinação é um mecanismo poderoso para combinar possibilidades e vasculhar todo o espaço de busca a fim de encontrar a melhor solução.

Depois que as configurações são selecionadas, o operador de recombinação é aplicado. As configurações são separadas em pares e a recombinação é realizada conforme a probabilidade predeterminada. Na recombinação, os pares selecionados trocam parcelas entre si, a partir de um ponto escolhido aleatoriamente para formar duas novas configurações.

Existem vários métodos de recombinação, a seguir serão apresentados alguns deles e, para ajudar na sua compreensão, ou seja, no entendimento de todo processo, observe as configurações dos pares abaixo:

1 - 1 0 1 0 0 0 0 0 1 0 0 1 1 1 0

2 - 1 1 0 1 1 1 0 1 1 0 0 1 0 0 0

**Recombinação de um simples ponto:** aleatoriamente, é escolhido o ponto de corte que servirá para trocar material genético entre as duas configurações. A posição escolhida deve ser comum entre o casal, mas não necessariamente deve ser igual para todos os casais selecionados.

**Recombinação de múltiplos pontos:** semelhante à recombinação com ponto simples, os locais são escolhidos aleatoriamente. Os locais de corte do indivíduo variam de 1 a  $N - 1$ , sendo  $N$  o comprimento total da configuração. Então, supondo-se que se esteja tratando do tipo com 4 pontos de corte, e que os escolhidos sejam os pontos 4, 6, 9 e 12, tem-se:

1 - 1 0 1 0 — 0 0 — 0 0 1 — 0 0 1 — 1 1 0

2 - 1 1 0 1 — 1 1 — 0 1 1 — 0 0 1 — 0 0 0

Resultado:

Primeiro Corte:

1 - 1 0 1 0 — 1 1 0 1 1 0 0 1 0 0 0

2 - 1 1 0 1 — 0 0 0 0 1 0 0 1 1 1 0

Segundo Corte:

1 - 1 0 1 0 1 1 — 0 0 1 0 0 1 1 1 0

2 - 1 1 0 1 0 0 — 0 1 1 0 0 1 0 0 0

Terceiro Corte:

1 - 1 0 1 0 1 1 0 0 1 — 0 0 1 0 0 0

2 - 1 1 0 1 0 0 0 1 1 — 0 0 1 1 1 0

Quarto Corte:

1 - 1 0 1 0 1 1 0 0 1 0 0 1 — 1 1 0

2 - 1 1 0 1 0 0 0 1 1 0 0 1 — 0 0 0

Portanto, o resultado final é:

1 - 1 0 1 0 1 1 0 0 1 0 0 1 1 1 0

2 - 1 1 0 1 0 0 0 1 1 0 0 1 0 0 0

**Recombinação Uniforme:** seguindo cada gene da primeira configuração, é verificado se ocorreu um evento com probabilidade de 50%. Caso afirmativo, ali será um ponto de corte; caso contrário, repete-se o procedimento para o gene posterior.

#### **4.2.2.5 Operador de Mutação**

A mutação é um operador genético que tem a função de introduzir características novas no indivíduo ou mesmo restaurar características que se perderam em operações como, por exemplo, a recombinação (SOARES, 1997).

Na teoria, a mutação é um evento que possui uma probabilidade  $p_m$  de acontecer ou não com cada gene de todas as configurações da população. Dependendo da probabilidade  $p_m$ , uma quantidade de genes sofrerá a ação da mutação. Serão escolhidos, aleatoriamente, os genes para serem mutados, sendo que eles poderão estar localizados em qualquer configuração da população, e em qualquer posição dentro da configuração. No caso da representação binária, se nos genes escolhidos for encontrado o valor zero, ele será substituído pelo valor um e vice-versa.

Segundo Vendramini (2007), o operador de mutação tem um papel fundamental na evolução da população, o de introduzir e de restaurar características, além de ajudar a evitar o problema de mínimo local ou convergência prematura. Contudo, deve-se tomar cuidado com a implementação, pois pode-se obter um custo computacional elevado.

No algoritmo genético simples, a mutação é considerada como um operador secundário, se comparada com a recombinação, e tem a finalidade de restaurar a perda de material genético que pode acontecer na população.

Depois da mutação, as configurações podem apresentar infactibilidades. É necessário formular estratégias, permitindo transformá-las em configurações factíveis. Uma alternativa é considerá-las todas factíveis e penalizar as infactibilidades na função objetivo a fim de que elas sejam eliminadas pelo operador de seleção.

#### **4.2.2.6 Parâmetros dos Algoritmos Genéticos**

Segundo Vendramini (2007), a cada geração, os algoritmos genéticos procuram as configurações com um melhor desempenho que as anteriores. A aplicação dos parâmetros de controle dos algoritmos genéticos (tamanho da população, taxa de recombinação e a taxa de mutação), bem como qual método utilizar para selecionar, recombinar e mutar, influenciam no desempenho do algoritmo genético. A seguir, é descrito, segundo Soares (1997), cada parâmetro de controle dos algoritmos genéticos:



**Número de Indivíduos da População ( $p$ ):** nos algoritmos genéticos, para que seja possível efetuar operações como recombinação e mutação, a configuração deve estar devidamente codificada. Uma população pequena possui amostragem insuficiente do espaço de busca do problema, podendo conduzir o algoritmo na direção de um ótimo local. Uma população grande contém uma quantidade bem representativa do espaço de busca, e também a perda da diversidade da população ocorrerá mais lentamente, possibilitando os algoritmos genéticos explorarem mais a informação existente. Entretanto, o número de cálculos de função objetivo, por geração, pode resultar em um tempo computacional inviável.

**Taxa de Recombinação ( $p_c$ ):** esse parâmetro controla a frequência com a qual o operador de recombinação é aplicado. A cada nova geração, provavelmente, uma parcela da população será recombinada.

**Taxa de Mutação ( $p_m$ ):** a mutação tem um papel diferente, porém não menos importante que a recombinação. As funções da mutação são inserir material genético novo e restaurar valores perdidos na recombinação, ou mesmo na mutação. Esse parâmetro controla a frequência com a qual o operador de mutação é aplicado.

Com base no programa de controle do algoritmo genético, um conjunto de parâmetros é idealizado para definir o tamanho da população, a taxa de recombinação e a de mutação. Esses parâmetros variam de problema para problema e, em grande parte, definem o sucesso do algoritmo na resolução do problema no qual foi aplicado.

#### **4.2.2.7 Critérios de Parada**

Segundo Vendramini (2007), existem vários critérios de parada, podendo ser implementado um único critério no problema ou, dependendo dele, mais de um, simultaneamente. Alguns critérios de parada dependem da evolução dos resultados do problema, outros são especificados previamente.

Os critérios de parada mais utilizados para finalizar a execução de um algoritmo genético são:

1. Quando atingir um número predeterminado de gerações.
2. Quando a melhor solução encontrada no problema assume um valor no mínimo equivalente a um valor previamente especificado.
3. Quando a incumbente não melhora durante um número especificado de iterações.
4. Quando as configurações da população perdem a diversidade, tornando-se muito parecidas umas com as outras, não evoluindo.

5. Usar um critério que depende do tipo de problema analisado. Em problemas reais, são relatados vários critérios de parada.

#### 4.3 ALGORITMO GENÉTICO DE CHAVES ALEATÓRIAS VICIADAS (GONÇALVES E RESENDE)

O algoritmo genético de chaves aleatórias viciadas (*Biased Random Key Genetic Algorithm* - BRKGA) foi proposto por Gonçalves e Resende (2013) e traz algumas diferenças em relação ao algoritmo genético tradicional. Entretanto, para entender seu funcionamento é preciso primeiro que seja definido o algoritmo genético de chaves aleatórias (*Random Key Genetic Algorithm* - RKGGA).

##### 4.3.1 Algoritmo Genético de Chaves Aleatórias

Introduzidos por Bean (1994), os algoritmos genéticos de chaves aleatórias, ou algoritmos de chaves aleatórias, foram apresentados, inicialmente, para resolver problemas de sequenciamento de tarefas.

Segundo Gonçalves e Resende (2013), em RKGGA, os cromossomos são representados por vetores com  $n$  elementos, os quais assumem valores gerados aleatoriamente no intervalo real  $[0, 1]$ . Portanto, uma proposta de solução é representada de forma codificada e deve ser decodificada usando-se um decodificador.

O decodificador é um algoritmo determinístico que utiliza como entrada um cromossomo, associando-o a uma solução do problema de otimização combinatória, para o qual um valor objetivo ou *fitness* pode ser calculado (GONÇALVES; RESENDE, 2013). Isso significa que, a partir de uma proposta de solução apresentada de forma codificada, encontra-se uma solução factível do problema original, assim como o valor da função objetivo que corresponde a essa proposta de solução. A forma e a estrutura do decodificador dependem do tipo de problema que se pretende resolver.

Segundo Gonçalves e Resende (2013), um RKGGA gera uma população de vetores de chaves aleatórias durante um número de gerações. A população inicial é formada de  $p$  vetores de  $n$  chaves aleatórias. Cada componente do vetor de solução, ou chave aleatória, é gerado, independentemente de forma aleatória no intervalo real  $[0, 1]$ . Após o *fitness* de cada indivíduo ser calculado pelo decodificador na geração  $g$ , a população é particionada em dois grupos de indivíduos: um grupo pequeno de  $p_e$  indivíduos de elite, ou seja, aqueles com os melhores

valores *fitness*, e um grupo formado pelo restante da população, o conjunto de  $p - p_e$ , indivíduos de não elite. Para evoluir da geração  $g$  para a  $g + 1$ , uma nova geração de indivíduos é produzida, o algoritmo RKGA usa elitismo. Todos os indivíduos de elite da população da geração  $g$  são copiados sem modificação para a população da geração  $g + 1$ . Como toda estratégia elitista, a ideia fundamental é preservar as melhores soluções e manter uma solução incumbente. Duas outras estratégias são usadas pelos RKGAs para completar a formação da geração  $g + 1$ , são elas: os operadores de mutação e de recombinação.

RKGAs implementam mutação introduzindo mutantes na população corrente. Um mutante é criado como sendo um vetor de chaves aleatórias, gerado da mesma maneira que um elemento da população inicial. A cada geração, um número pequeno de mutantes  $p_m$  é introduzido na população. Os mutantes substituem a estratégia de mutação usada no algoritmo genético tradicional (GONÇALVES; RESENDE, 2013).

A população das próximas gerações será composta por indivíduos de elite ( $p_e$ ), indivíduos mutantes ( $p_m$ ) e indivíduos produzidos pelo processo de recombinação. Com  $p_e + p_m$  indivíduos representando a população da geração  $g + 1$ ,  $p - p_e - p_m$ , indivíduos adicionais precisam ser gerados para completar os  $p$  indivíduos que formam a população da geração  $g + 1$ . Isso é realizado produzindo  $p - p_e - p_m$  soluções descendentes por meio do processo de recombinação.

Nos algoritmos RKGA e BRKGA, não existe o operador de seleção convencional, como apresentado na seção 4.2.2.4 do algoritmo genético tradicional. É exatamente nesse ponto que os dois algoritmos de chaves aleatórias se diferenciam, pois um BRKGA se distingue do RKGA na forma como os pais são selecionados para a recombinação.

### 4.3.2 Algoritmo Genético de Chaves Aleatórias Viciadas

No RKGA de Bean (1994), os pais são selecionados aleatoriamente dentro do conjunto da população corrente. Assim, para implementar a recombinação, há dois elementos da geração  $g$ , e esses elementos são submetidos à recombinação para gerar apenas um descendente, o qual será incorporado na população da geração  $g + 1$ . Essa estratégia permite que soluções de qualidade, diversificadas, sejam escolhidas aleatoriamente, já que não é usado o valor da função objetivo para escolher as soluções geradoras de novas soluções.

Já no BRKGA, de Gonçalves e Resende (2013), cada elemento é gerado combinando um indivíduo selecionado aleatoriamente da partição de elite da população corrente. Para a

seleção do segundo indivíduo, têm-se duas estratégias de obtenção: na primeira estratégia, obtém-se o segundo indivíduo selecionando-o da partição não elite da população corrente; já na segunda, selecionando-o da população corrente, o que inclui tanto a partição elite quanto a não elite.

Deve-se observar que se encontra de forma implícita o operador de seleção. A primeira estratégia sempre escolhe uma solução de elite. Enquanto que, na segunda, sempre existirá um indivíduo da partição de elite, e outro que poderá ser de elite ou não. Como as soluções de elite representam um número reduzido de elementos, então a participação das soluções de elite deve ser muito intensa na geração de novas soluções no algoritmo BRKGA, quando comparado ao algoritmo RKGA.

A palavra viciada, no algoritmo BRKGA, incorporada pelos autores Gonçalves e Resende (2013), é utilizada para mostrar que a recombinação não é puramente aleatória. Isto é, uma das soluções geradoras deve ser necessariamente de elite (viciada), em contraposição à proposta original, na qual as duas soluções geradoras podem ser do conjunto da população.

Repetição na seleção de um par é permitida e, portanto, um indivíduo pode produzir mais de um descendente na mesma geração. Assim como nos RKGAs, os BRKGAs implementam recombinação uniforme parametrizada (SPEARS; DEJONG, 1991).

Nesse tipo de recombinação, tem-se a probabilidade  $q_e$  que servirá como um parâmetro balizador para que se saiba se o componente  $n$ , do vetor descendente  $C$ , será herdado do pai  $A$  ou do pai  $B$ . Deve-se lembrar que  $n$  denota o número de componentes em um vetor de solução de um indivíduo. Para  $i = 1, \dots, n$ , o componente  $i$ -ésimo de  $c(i)$  do vetor descendente  $C$  assume o valor do  $i$ -ésimo componente  $a(i)$  do pai de elite  $A$  com probabilidade  $q_e$  ou o valor do  $i$ -ésimo componente  $b(i)$  do pai  $B$  oriundo do conjunto da população com probabilidade  $1 - q_e$ .

O valor usado para  $q_e$  é elevado e tipicamente é escolhido  $q_e = 0,7$ . Assim, em cada caso, existe uma probabilidade de 0,7 para que o valor armazenado em  $A$  seja copiado no descendente  $C$  que está sendo gerado. Essa estratégia permite que a maioria dos elementos copiados em  $C$  sejam armazenados na solução de elite, deixando claro os motivos pelos quais é gerado apenas um descendente, ao contrário do que é realizado no algoritmo genético tradicional. A Tabela 3, que se encontra ao final da seção 3.4.3 deste trabalho, ilustra este tipo de recombinação.

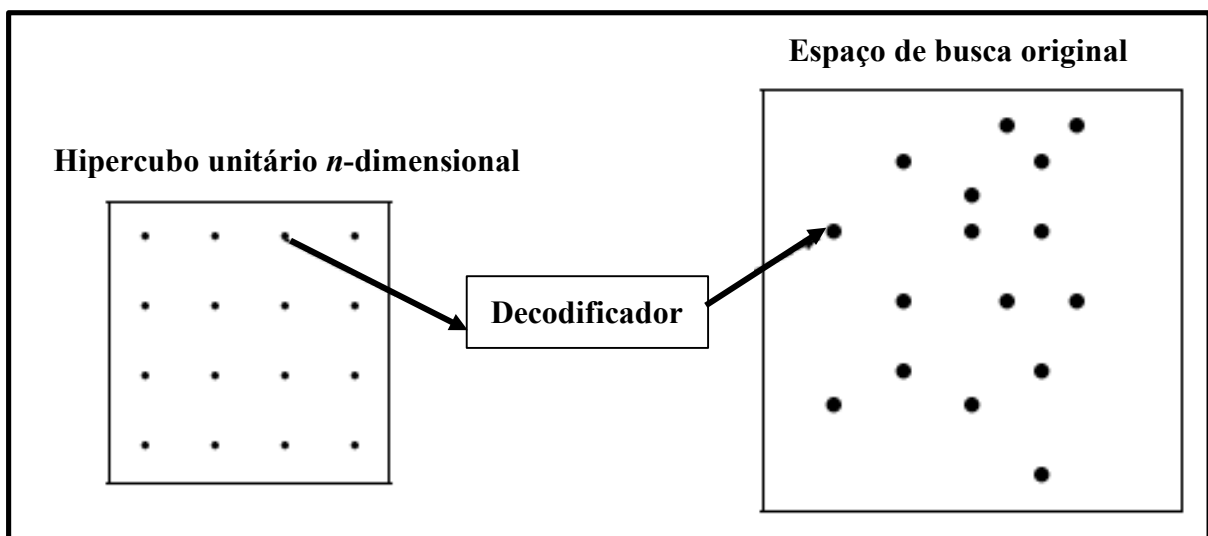
Quando a próxima população é gerada, os valores *fitness* desses indivíduos são calculados, usando o decodificador, e a população é particionada em indivíduos de elite e não elite para começar uma nova geração, até que se verifique satisfeito o critério de parada.

Segundo Gonçalves e Resende (2013), um algoritmo BRKGA procura o espaço de solução do problema de otimização combinatória indiretamente, buscando o hipercubo contínuo  $n$ -dimensional e, ainda, usando o decodificador para mapear as soluções no hipercubo para soluções no espaço de solução do problema de otimização combinatória onde o *fitness* é avaliado.

Para especificar um algoritmo genético de chaves aleatórias viciadas, simplesmente, precisa-se especificar como as soluções são codificadas e decodificadas e como seus correspondentes valores *fitness* são calculados. A Figura 26 mostra a função do decodificador.

A meta-heurística BRKGA, na verdade, apresenta uma estrutura geral para resolver problemas muito variados. Essa estrutura apresenta duas parcelas claramente diferenciadas como se mostra na Figura 27. Segundo Gonçalves e Resende (2013), existe uma parcela que é independente do problema e, portanto, não precisa conhecer o problema que está sendo resolvido. Essa parcela limita a busca a um espaço definido por um hipercubo unitário  $n$ -dimensional. Assim, a única conexão com o problema que está sendo resolvido é por meio do decodificador, que recebe uma proposta de solução codificada na forma de chaves aleatórias. A partir dessa solução, encontra-se a solução no espaço de busca original, assim como o valor da função objetivo. Portanto, para definir as características específicas de um algoritmo BRKGA, precisa-se apenas definir a forma de codificação na forma de chaves aleatórias e o decodificador.

**Figura 26** - Decodificador que Procura a Solução no Espaço de Busca Original



Fonte: Adaptado de Gonçalves e Resende (2013)

O algoritmo BRKGA apresenta poucos parâmetros que precisam ser escolhidos e

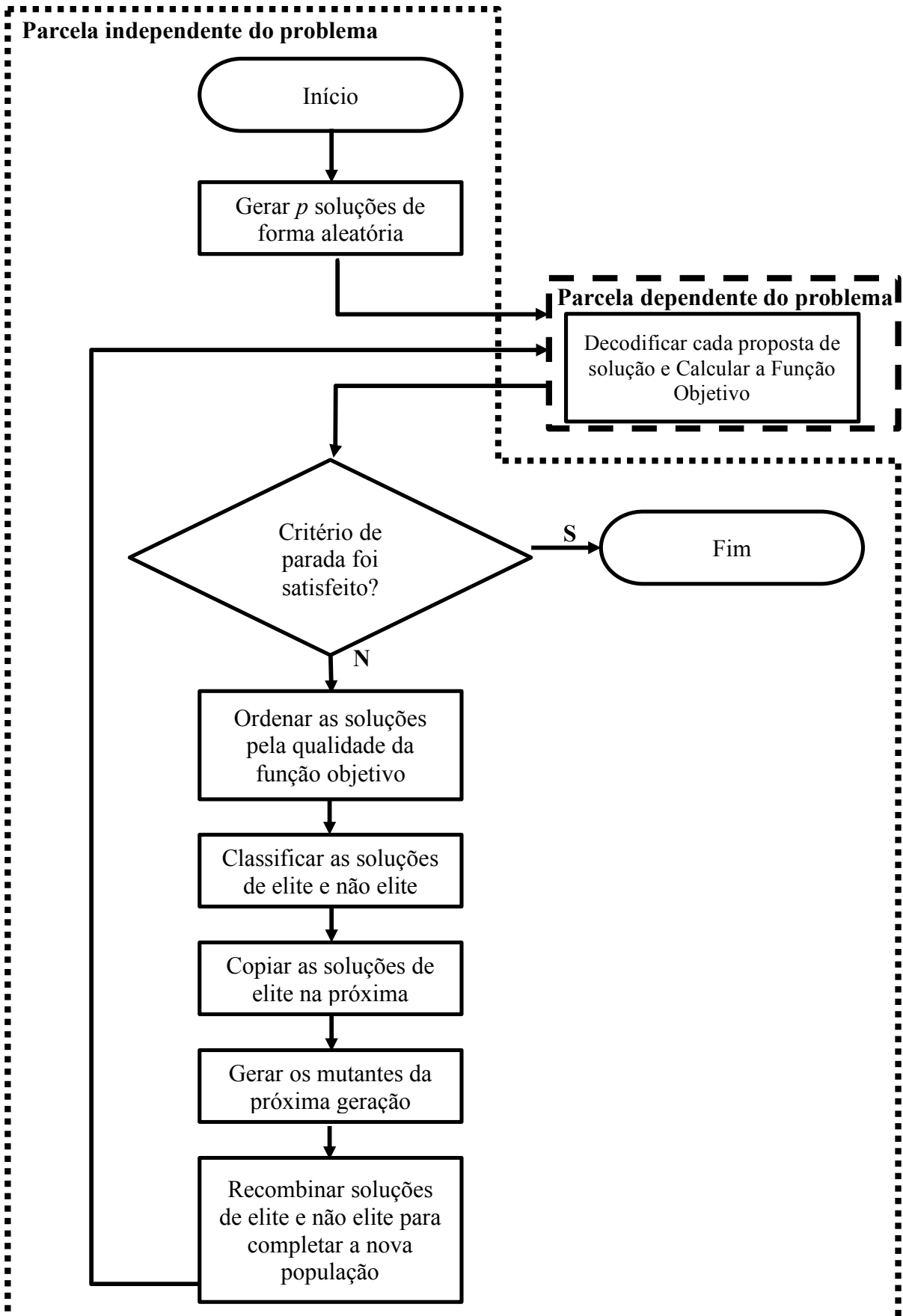
calibrados. Segundo Gonçalves e Resende (2013), esses parâmetros são a dimensão do vetor de codificação de chaves aleatórias ( $n$ ), o tamanho da população ( $p$ ), o número das soluções consideradas de elite ( $p_e$ ), o número das soluções mutantes ( $p_m$ ) e a probabilidade ( $q_e$ ) de que um descendente incorpore a informação armazenada na solução geradora de elite. Como em toda meta-heurística, calibrar adequadamente esses parâmetros já representa um novo problema; também, como em toda meta-heurística, esses parâmetros são calibrados de forma empírica levando em consideração as características físicas reais do problema sob análise. Na Tabela 5, a seguir, são apresentados valores recomendados para esses parâmetros segundo Gonçalves e Resende (2013).

**Tabela 5** - Valores Recomendados dos Parâmetros do Algoritmo BRKGA

Parâmetro	Especificação	Valores Recomendados
$p$	Tamanho da população	$p = a$ , onde $1 \leq a \in R$ é uma constante que depende da dimensão de uma proposta de solução codificada pelo vetor de chaves aleatórias
$p_e$	Tamanho das soluções de elite	$0,10p \leq p_e \leq 0,25p$
$p_m$	Tamanho das soluções mutantes	$0,10p \leq p_m \leq 0,30p$
$q_e$	Probabilidade de herdar a informação da solução geradora de elite	$0,5 \leq q_e \leq 0,8$

Fonte: Gonçalves e Resende (2013)

Figura 27 - Diagrama de Fluxo do Algoritmo BRKGA



Fonte: Adaptado de Gonçalves e Resende (2013) pela autora.

#### 4.4 ALGORITMO GENÉTICO DE CHAVES ALEATÓRIAS VICIADAS ESPECIALIZADO

A estrutura tradicional do algoritmo genético (seleção, recombinação e mutação) é mantida no algoritmo genético de chaves aleatórias viciadas especializado, proposto neste trabalho, porém o algoritmo especializado acrescenta alguns processos que se fizeram necessários para melhor atender ao problema a ser solucionado.

A base para o algoritmo especializado foi o algoritmo genético de chaves aleatórias viciadas, proposto por Gonçalves e Resende (2013), em que o controle da diversidade é mais acentuado do que o algoritmo genético tradicional. Porém, a autora deste trabalho, com base no que Gonçalves e Resende (2013) propuseram, acrescentou algumas modificações, tornando o algoritmo, aqui proposto, ainda mais especializado para o problema de corte bidimensional não guilhotinado.

##### 4.4.1 Funcionamento do Algoritmo Genético de Chaves Aleatórias Viciadas Especializado

A seguir, tem-se a descrição detalhada do funcionamento do algoritmo genético de chaves aleatórias viciadas especializado.

- A) O algoritmo especializado inicia seu processamento gerando uma população inicial de tamanho  $p$ . Os indivíduos dessa população inicial são compostos por valores reais, gerados aleatoriamente no intervalo  $[0, 1]$ . Cada indivíduo, ou proposta de solução codificada, tem, em seu interior, duas parcelas: a primeira, de 1 até  $n$ , que representa a ordem em que as peças serão cortadas na placa, chama-se essa parcela de BPS; a segunda parcela,  $n + 1$  até  $2n$ , que representa a orientação de como essas peças serão cortadas na placa, chama-se essa parcela de VBO (GONÇALVES; RESENDE, 2013), conforme já foi detalhado anteriormente na seção 3.4 deste trabalho. Na Figura 28, representa-se uma proposta de solução codificada, inicialmente utilizada pelo algoritmo especializado.

**Figura 28** - Proposta de Solução Inicial Codificada Utilizada pelo Algoritmo Genético de Chaves Aleatórias Viciadas Especializado

$$\text{Proposta de solução inicial codificada} = \underbrace{(p_1, p_2, \dots, p_n)}_{\text{Sequência de peças cortadas}} \underbrace{(o_{n+1}, o_{n+2}, \dots, o_{2n})}_{\text{Orientação das Peças}}$$

Fonte: Elaborado pela autora.



B) Em seguida, o algoritmo especializado faz a decodificação de cada indivíduo da população. A decodificação ocorre nos mesmos moldes propostos por Gonçalves e Resende (2013). A parcela BPS da proposta de solução é decodificada, classificando os seus valores em ordem crescente, ou seja, dispondo dos seguintes valores na parcela BPS: 0,45 representando a peça 1; 0,76, a peça 2; 0,49, a peça 3. O decodificador ordenará da seguinte maneira: 0,45; 0,49; 0,76. Isso significa que a peça 1, representada pelo número 0,45, será a primeira cortada na placa. Logo em seguida, a peça 3, representada pelo número 0,49. Por fim, a peça 2, representada pelo número 0,76. A Figura 22, mostrada na seção 3.4 deste trabalho, retrata um exemplo do processo de decodificação na parcela *BPS*. A decodificação da parcela *VBO* da proposta de solução é realizada por  $i = 1, \dots, n$ , usando-se a seguinte expressão:

$$VBO_i = \begin{cases} 0 & \text{se } 0 \leq o_{n+i} \leq \frac{1}{2}, \\ 1 & \text{se } \frac{1}{2} < o_{n+i} \leq 1. \end{cases}$$

Se encontrados valores entre 0 e 0,5, a orientação da peça não será alterada, ou seja, ela será cortada na placa, seguindo suas medidas originais, seu comprimento e sua altura. O *VBO*, nessas posições, assume o valor 0 (zero). Contudo, se na parcela *VBO* forem encontrados valores acima de 0,5 até 1, a peça sofrerá uma rotação, isto é, o que era comprimento passa a ser altura, e o que era altura passa a ser comprimento. Nessas posições, o *VBO* assume o valor 1 (um). Portanto, a orientação das peças a serem cortadas, na placa, seguem o resultado do decodificador da parcela *VBO* da proposta de solução.

C) Logo após a decodificação, um Algoritmo Heurístico Construtivo (AHC) é acionado para transformar cada proposta de solução da população corrente em um padrão de corte. O AHC do algoritmo especializado funciona seguindo os seguintes passos:

- 1) Criar uma lista de retângulos livres existentes na placa, ou seja, todos os espaços que não foram utilizados para cortar uma peça dentro da placa são considerados livres e irão compor essa lista.
- 2) Verificar qual é a primeira peça a ser cortada na placa. Essa informação é obtida verificando-se a parcela *BPS* da proposta de solução.

- 3) Criar um bloco retangular (ALVAREZ-VALDES; PARREÑO; TAMARIT, 2007) com peças iguais à primeira a ser cortada, respeitando seu limitante superior. Já que cada peça não é necessariamente única, ela tem uma quantidade mínima e uma quantidade máxima de cópias que pode ser cortada na placa.
- 4) Observar que a origem da placa é seu canto inferior esquerdo e tendo sempre a origem como ponto de referência para o corte dos blocos de peças, cortar o bloco criado no passo 3, na origem da placa, seguindo a orientação da primeira peça. Essa informação é obtida verificando-se a parcela VBO da proposta de solução. Todos os blocos seguintes devem sempre ser cortados o mais próximo da origem (LAI; CHAN, 1997).
- 5) Atualizar a lista dos retângulos livres após o bloco ser cortado na placa. Para obter os retângulos livres, a autora deste trabalho propõe um escaneamento da placa, de cima para baixo e da esquerda para a direita. Após a obtenção dos retângulos livres, aplicar o processo de diferença e eliminação proposto por Lai e Chan (1997), detalhado na seção 3.2 deste trabalho.
- 6) Verificar se há retângulos livres que possam ser utilizados pela próxima peça a ser cortada na placa, respeitando a ordem das peças, estabelecida pela parcela BPS da proposta de solução, e seguindo a orientação da peça, estabelecida pela parcela VBO da proposta de solução. Se não existir retângulo livre disponível para cortar a próxima peça respeitando a sua orientação, alterar a orientação da peça, girando-a. Verificar, ainda, se dessa maneira existe algum retângulo livre que possa ser utilizado para o corte da peça. Se mesmo assim não houver retângulo livre disponível, essa peça não será cortada na placa e passa-se para a próxima peça, respeitando-se a ordem em que elas aparecem na parcela BPS da proposta de solução.
- 7) Escolher o retângulo que se encontra mais próximo da origem da placa caso haja mais de um retângulo livre disponível para o corte da peça, (LAI; CHAN, 1997).
- 8) Formar um bloco retangular da peça (ALVAREZ-VALDES; PARREÑO; TAMARIT, 2007), respeitando as dimensões do retângulo livre escolhido e a quantidade de cópias da peça disponíveis para corte na placa.
- 9) Cortar, na placa, o bloco formado no passo 8.

- 10) Voltar ao passo 5 e repetir a sequência de passos do 5 ao 9 até o final das peças encontradas na parcela BPS da proposta de solução. Lembrando que essa sequência de passos deve ser seguida para todo indivíduo da população corrente.
- D) Ao final do AHC do algoritmo especializado, tem-se, para cada indivíduo da população corrente, um padrão de corte formado por: sequência das peças cortadas, quantidade de cópias da peça utilizada em cada bloco cortado, orientação que foi utilizada no momento do corte de cada bloco e, ainda, coordenadas onde cada bloco foi cortado. Essa é uma proposta de representação de solução do problema que a autora do trabalho sugere e que foi detalhada na seção 3.5 deste trabalho.
- E) Na sequência, os valores *fitness* desses indivíduos são calculados. Para esse cálculo, considera-se melhor o indivíduo que encontra o maior valor depois de somados os valores de todas as peças que foram cortadas na placa. Então, a partir do valor *fitness* de cada indivíduo, o grupo de elite é identificado, classificando-se a população corrente como elite e não elite (GONÇALVES; RESENDE, 2013).
- F) Para dar origem à próxima geração, os indivíduos classificados como elite,  $p_e$ , são copiados na íntegra para a geração  $g + 1$ . Mutantes,  $p_m$ , são gerados assim como os indivíduos da população inicial foram gerados e acrescentados à geração  $g + 1$ , seguindo algoritmo BRKGA proposto por Gonçalves e Resende (2013).
- G) Os outros  $p - p_e - p_m$  indivíduos que faltam para compor a próxima geração, são oriundos do processo de recombinação. Cada indivíduo é gerado combinando um pai selecionado aleatoriamente da partição de elite da população corrente, e outro pai selecionado da mesma maneira, porém do conjunto da população corrente, o que inclui elite e não elite. Depois de escolhidas as soluções geradoras, elas serão recombinadas, criando-se somente um descendente para a próxima geração populacional  $g + 1$ . A recombinação ocorre seguindo a proposta de Gonçalves e Resende (2013), o valor do parâmetro  $q_e$ , utilizado no algoritmo especializado, foi de 0,7. Esse processo de recombinação já foi descrito anteriormente, especificamente na seção 3.4 deste trabalho.
- H) O algoritmo especializado retorna ao passo B) e repete a sequência de B) a H), até que o critério de parada seja satisfeito. No caso do algoritmo especializado, o critério de parada utilizado foi um número preestabelecido de gerações.

#### 4.4.2 Formação dos Blocos

No algoritmo proposto neste trabalho, constrói-se o bloco com peças iguais, dando prioridade ao sequenciamento delas no sentido horizontal. Tal procedimento de agrupá-las em blocos não foi utilizada no algoritmo BRKGA, proposto por Gonçalves e Resende (2013), portanto trata-se de uma melhoria, proposta pela autora, aplicada no algoritmo genético de chaves aleatórias viciadas especializado.

As peças  $p_i$  são alocadas uma ao lado da outra até que não se tenha mais espaço disponível dentro do retângulo livre escolhido para alocar mais peças, ou até quando a disponibilidade das peças  $p_i$  se esgotarem. Somente após as peças  $p_i$  serem alocadas, uma ao lado da outra, e havendo a disponibilidade de mais peças desse mesmo tipo serem utilizadas na construção do bloco, procura-se alocar peças  $p_i$  acima das peças  $p_i$  já alocadas no retângulo livre.

As peças, na parte de cima, somente se efetivarão nessa posição se houver a mesma quantidade disponível de peças  $p_i$  utilizadas na parte de baixo do bloco. E, assim, ocorrerá sucessivamente até que a quantidade de peças  $p_i$ , utilizadas na construção do bloco, seja  $\leq Q_i$ . Dessa maneira, garante-se um bloco com forma geométrica retangular ou quadrada.

#### 4.4.3 Escaneamento de Retângulos Livres

Durante a pesquisa realizada sobre o problema de corte bidimensional não guilhotinado, verificou-se que a identificação de retângulos livres, na placa, é de suma importância para o sucesso do algoritmo na resolução do problema, pois a partir da produção de retângulos livres é que se escolhe qual retângulo será utilizado no próximo corte dentro da placa.

Lai e Chan (1997), por exemplo, dedicaram um tópico, em seu artigo, exclusivamente para explicar a técnica de Geração de Intervalos que resulta nos retângulos livres, dada a importância do assunto.

A proposta deste trabalho consiste em escanear a placa mapeando os espaços que estão disponíveis para novos cortes. O procedimento age como um escâner, mapeando a placa de cima para baixo e da esquerda para a direita. Esta é uma outra proposta de melhoria, utilizada pela autora, para ser aplicada ao algoritmo genético de chaves aleatórias viciadas especializado. Além disso, é uma contribuição inédita do trabalho, já que o tipo de escaneamento feito na placa mostra-se diferente daqueles encontrados na literatura especializada.

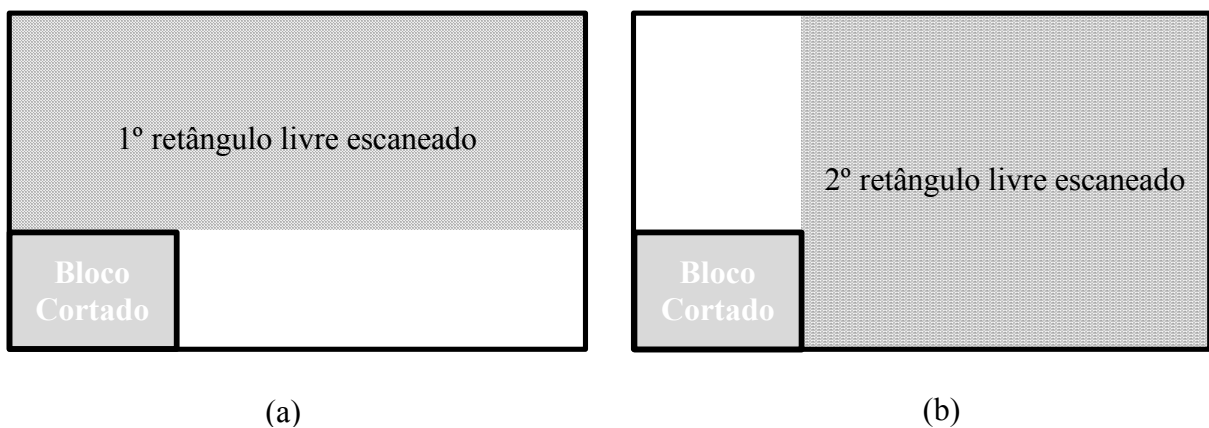
Quando um espaço é identificado como livre, é adicionado a ele todos os espaços livres e sequenciais que estão logo abaixo dele. Posteriormente, seguindo o mesmo procedimento, adicionam-se os espaços livres que estão à sua direita, obedecendo à delimitação de altura realizada pela primeira coluna do escaneamento, dando, assim, origem a um retângulo livre. Usando-se essa lógica, faz-se o escaneamento de cima para baixo.

Utilizando-se dessa mesma lógica, emprega-se o escaneamento da esquerda para a direita. Quando um espaço é identificado como livre, são adicionados a ele todos os espaços livres e sequenciais que estão à sua direita. Em seguida, adotando-se o mesmo raciocínio, adicionam-se os espaços livres que estão logo abaixo, obedecendo à delimitação de comprimento (ou altura) realizada pela primeira linha do escaneamento, dando, assim, origem a um retângulo livre.

Verifica-se que um espaço livre pode ser mapeado em mais de um retângulo livre, dando a liberdade para o algoritmo de resolução do problema de corte bidimensional não guilhotinado escolher qual retângulo livre é melhor para ele naquele momento.

Na Figura 29, ilustra-se como o procedimento de escaneamento de retângulos livres funciona.

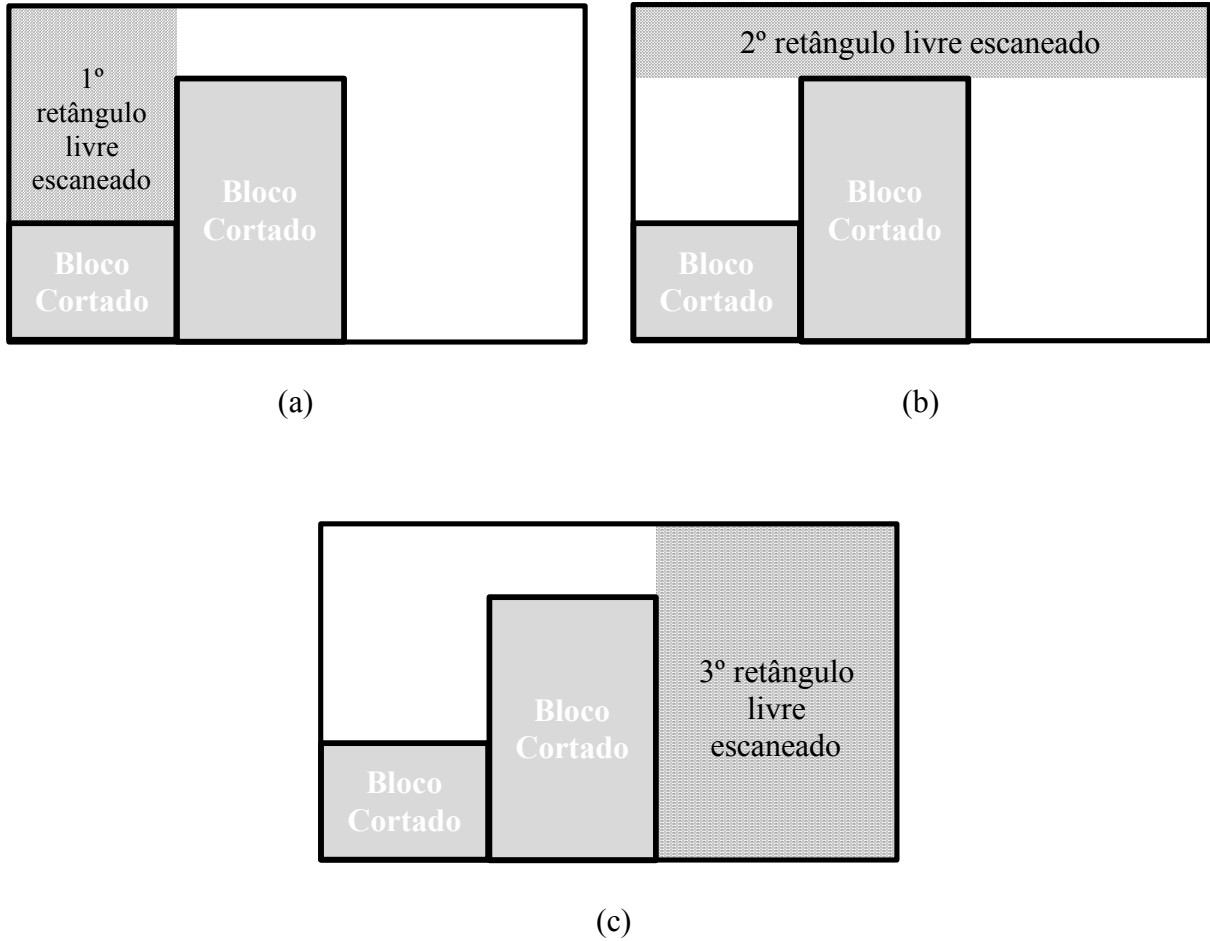
**Figura 29** - Escaneamento de Retângulos Livres. (a) Primeiro Retângulo Livre Escaneado, (b) Segundo Retângulo Livre Escaneado



Fonte: Elaborado pela autora.

Ao cortar o segundo bloco de peça, um novo escaneamento é realizado seguindo os mesmos procedimentos. Veja, na Figura 30, como ficaria o escaneamento de retângulos livres neste caso.

**Figura 30** - Escaneamento de Retângulos Livres Depois do Corte da Segunda Peça. (a) Primeiro Retângulo Livre Escaneado, (b) Segundo Retângulo Livre Escaneado e (c) Terceiro Retângulo Livre Escaneado



Fonte: Elaborado pela autora.

## 5 ALGORITMO RVNS ESPECIALIZADO PARA O PROBLEMA DE CORTE BIDIMENSIONAL NÃO GUILHOTINADO

Neste capítulo, o leitor encontrará uma outra meta-heurística proposta pela autora para a resolução do problema de corte bidimensional não guilhotinado. Essa nova proposta tem como base a meta-heurística de Busca em Vizinhança Variável Reduzida (RVNS) e, ainda, traz quatro estruturas de vizinhança específicas para o problema em estudo, tornando o algoritmo RVNS, proposto pela autora, especializado em resolver o problema de corte bidimensional não guilhotinado.

O capítulo inicia com uma introdução sobre a meta-heurística de Busca em Vizinhança Variável (VNS), o que pode ser observado na seção 5.1. Já na seção 5.2, mostram-se os fundamentos de VNS; na seção 5.3, o algoritmo VND; na seção 5.4, o algoritmo RVNS; na seção 5.5, o algoritmo BVNS; na seção 5.6, o algoritmo GVNS. Por fim, na seção 5.7, é descrito o algoritmo RVNS, especializado para o problema de corte bidimensional não guilhotinado. Esse algoritmo especializado é a segunda meta-heurística proposta, neste trabalho, para a resolução do problema em estudo.

### 5.1 INTRODUÇÃO À META-HEURÍSTICA DE BUSCA EM VIZINHANÇA VARIÁVEL

A Busca em Vizinhança Variável, VNS (do inglês *Variable Neighborhood Search*), é uma meta-heurística que foi proposta em meados da década de 90, por Mladenović (1995); Hansen e Mladenović (1997).

Segundo Hansen e Mladenović (2001), a Busca em Vizinhança Variável (VNS) é uma meta-heurística recente, ou uma estrutura para construir heurísticas, que explora sistematicamente a ideia de troca de estruturas de vizinhança, tanto na descida para ótimos locais, quanto para escapar de vales que os contêm. A meta-heurística utiliza o mecanismo de troca de vizinhança em conjunto com um algoritmo de busca local, ou seja, a meta-heurística de Busca em Vizinhança Variável é uma extensão de um algoritmo de busca local que utiliza a estratégia de mudança de tamanho da vizinhança para sair de soluções ótimas locais.

A meta-heurística VNS explora vizinhanças cada vez mais distantes da solução incumbente corrente e muda da incumbente corrente para uma nova se, e somente se, uma melhoria for obtida. Na maioria dos casos, as características favoráveis de uma solução incumbente serão mantidas e usadas para obter soluções vizinhas promissoras (HANSEN; MLADENOVIĆ, 2001).

A meta-heurística VNS, mais especificamente o algoritmo RVNS, foi escolhido pela autora por se tratar de uma meta-heurística que apresenta excelentes resultados em trabalhos da área de pesquisa operacional, porém é pouco utilizada no problema de corte bidimensional não guilhotinado.

A boa performance da meta-heurística VNS está ligada ao fato dela apresentar características desejáveis a uma meta-heurística. Hansen e Mladenović (2003b) analisaram o quanto a do tipo VNS é capaz de se ajustar a estas características. A lista a seguir foi adaptada de Hansen e Mladenović (2003b):

- i. Simplicidade: a meta-heurística deve estar baseada num princípio simples e claro, que possa ser amplamente aplicável;
- ii. Precisão: os passos da meta-heurística devem ser formulados com precisão;
- iii. Coerência: todas as etapas de uma heurística para um problema específico devem derivar naturalmente do princípio da meta-heurística;
- iv. Eficiência: os procedimentos devem fornecer soluções ótimas ou quase ótimas para a maioria dos problemas reais;
- v. Eficácia: os algoritmos devem empregar um esforço computacional moderado para fornecer soluções ótimas ou quase ótimas;
- vi. Robustez: o desempenho dos algoritmos deve ser consistente em uma variedade de casos, isto é, devem dar boas soluções para várias instâncias do problema;
- vii. Facilidade de uso: deve se expressar claramente, ser fácil de entender e de usar. Isso implica que devem ter poucos, ou idealmente, nenhum parâmetro;
- viii. Inovação: os princípios das meta-heurísticas, e/ou eficiência e eficácia das heurísticas derivadas delas, devem levar a novas aplicações;
- ix. Generalidade: a meta-heurística deve obter bons resultados para uma grande variedade de problemas;
- x. Interatividade: a meta-heurística deve permitir ao usuário incorporar seu conhecimento, de forma a melhorar o processo de solução;
- xi. Multiplicidade: a meta-heurística deve ser capaz de apresentar várias soluções quase ótimas, dentre as quais o usuário possa escolher.

A meta-heurística VNS permite não somente observar que soluções de boa qualidade foram obtidas, mas também entender o porquê de tal acontecimento. Além disso, o algoritmo VNS ajuda a elaborar melhorias para casos em que as soluções obtidas são de baixa qualidade, geralmente na forma de novas vizinhanças.

A meta-heurística VNS trabalha com métodos de busca local que realizam uma



sequência de modificações locais em uma solução inicial, melhorando o valor da função objetivo, até que um ótimo local seja encontrado.

Portanto, um algoritmo de busca local, em geral, segue os passos apresentados a seguir.

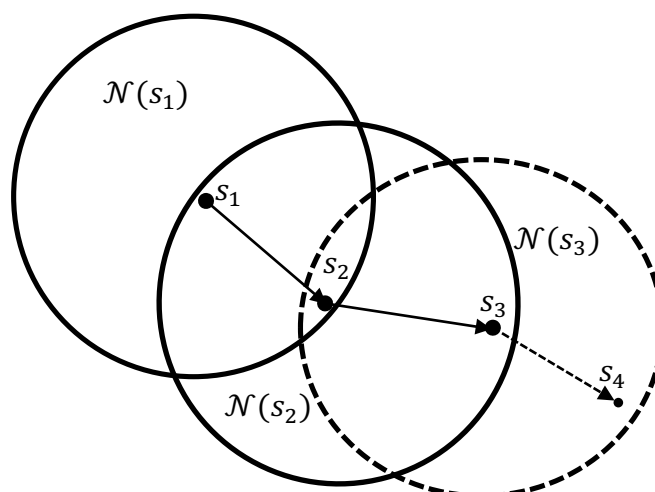
**Passo 1:** Gerar uma solução inicial  $s$ ;

**Passo 2:** Aplicar uma busca local para encontrar a melhor solução vizinha  $s'$ ;

**Passo 3:** Se  $s'$  é melhor que  $s$  então faça  $s = s'$  e vá para o Passo 2, senão pare.

A cada iteração, uma solução melhorada  $s'$  da vizinhança  $\mathcal{N}(s)$  da solução corrente  $s$  é obtida, até que nenhuma melhora possa ser encontrada. A Figura 31 ilustra o processo de busca de um algoritmo de busca em vizinhança. Partindo de um ponto inicial  $s_1$ , o algoritmo analisa toda a vizinhança de  $s_1$ ,  $\mathcal{N}(s_1)$ , encontrando uma solução de melhor qualidade,  $s_2$ . A vizinhança de  $s_2$  é, então, analisada, obtendo uma solução de melhor qualidade,  $s_3$ . Ao analisar a vizinhança de  $s_3$ ,  $\mathcal{N}(s_3)$ , verifica-se que a melhor solução nessa vizinhança,  $s_4$ , é de pior qualidade que  $s_3$  e, assim, o processo termina. Segundo Glover e Kochenberger (2003), existem várias meta-heurísticas que estendem esse esquema de diferentes formas, evitando que o processo fique preso em um ótimo local de baixa qualidade.

**Figura 31** - Processo de Busca de um Algoritmo de Busca em Vizinhança



Fonte: Adaptado de Possagnolo (2015) pela autora.

## 5.2 FUNDAMENTOS DE VNS

A Busca em Vizinhança Variável é baseada num princípio simples: explorar o espaço de soluções por meio de trocas sistemáticas de estruturas de vizinhança durante o processo de busca (HANSEN; MLADENOVIC, 2001).

A VNS, ao contrário de muitas meta-heurísticas baseadas em busca local, não permite degradação na função objetivo para realizar um movimento, mas explora, a partir de uma solução inicial, uma sequência crescente de vizinhanças, e só realiza a transição da solução incumbente corrente para uma nova solução incumbente se esta nova for melhor que a corrente.

Segundo Hansen e Mladenović (2001), a meta-heurística VNS explora sistematicamente as três propriedades seguintes:

**Fato 1:** um ótimo local com relação a uma estrutura de vizinhança não é necessariamente o mesmo ótimo local para outra estrutura de vizinhança;

**Fato 2:** um ótimo global é um mínimo local em relação a todas as possíveis estruturas de vizinhança;

**Fato 3:** para muitos problemas, ótimos locais com relação a uma ou várias estruturas de vizinhança são relativamente próximos uns dos outros.

Os Fatos acima, de 1 a 3, sugerem o uso de várias estruturas de vizinhança nas buscas locais para abordar um problema de otimização, ou seja, a ideia é definir um conjunto de estruturas de vizinhanças que possam ser utilizadas de forma determinística, aleatória ou determinística e aleatória. Essas formas de utilizar as estruturas de vizinhanças produzem algoritmos VNS de desempenhos diferentes.

O Fato 3 é particularmente importante na formulação de um algoritmo VNS, o qual é de caráter empírico, implica que uma solução ótima local fornece informações importantes em relação ao ótimo global, especialmente se a solução ótima local for de excelente qualidade. Existe também a observação empírica de que as soluções ótimas locais, geralmente, estão concentradas em regiões específicas do espaço de busca. Se as soluções ótimas locais estivessem uniformemente distribuídas no espaço de busca, todas as heurísticas baseadas em uma única vizinhança se tornariam ineficientes (HANSEN; MLADENOVIĆ, 2001).

Portanto, se for encontrado um ótimo local da região em que se encontra o ótimo global, então uma meta-heurística tipo VNS possui grandes chances de encontrar esse ótimo global. Por outro lado, caso o ótimo global encontre-se em outra região, a única possibilidade de encontrá-lo é implementando um processo de diversificação. Por esse motivo, um equilíbrio entre intensificação e diversificação, no processo de busca, pode ser importante em uma meta-heurística.

Existem várias propostas de algoritmos VNS que podem ser usadas de forma independente ou integrada em estruturas VNS mais complexas. A seguir, são apresentados quatro tipos de algoritmos VNS: algoritmo VND, algoritmo RVNS, algoritmo BVNS, e algoritmo GVNS.

### 5.3 VNS DE DESCIDA (VND)

A meta-heurística VND (do inglês *Variable Neighborhood Descent*) é baseada no Fato 1, citado anteriormente, ou seja, um ótimo local para uma estrutura de vizinhança não é necessariamente um ótimo local para outra estrutura de vizinhança (o ótimo local  $\mathbf{s}'$  na vizinhança  $\mathcal{N}_1(\mathbf{s})$  não é necessariamente igual ao ótimo local  $\mathbf{s}''$  na vizinhança  $\mathcal{N}_2(\mathbf{s})$ ).

O algoritmo VND, apresentado por Hansen e Mladenović (2001), assume a estrutura mostrada na Figura 32.

**Figura 32** - Algoritmo VND

**Inicialização:** selecione o conjunto de estruturas de vizinhança  $\mathcal{N}_k$ ,  $l = 1, \dots, k_{max}$ , que será utilizado na descida; encontre uma solução inicial  $\mathbf{s}$ ;

**Repita** os passos seguintes até que nenhuma melhoria na solução seja obtida:

- (1) Faça  $l \leftarrow 1$ ;
- (2) **Repita** os passos a seguir até  $k = k_{max}$ :
  - (a) **Exploração da vizinhança:** encontre o melhor vizinho  $\mathbf{s}'$  de  $\mathbf{s}$  ( $\mathbf{s}' \in \mathcal{N}_1(\mathbf{s})$ );
  - (b) **Mover ou não:** se a solução  $\mathbf{s}'$  é melhor que a incumbente  $\mathbf{s}$ , faça  $\mathbf{s} \leftarrow \mathbf{s}'$  e continue a busca em  $\mathcal{N}_1(k \leftarrow 1)$ ; caso contrário faça  $k \leftarrow k + 1$ .

Fonte: Hansen e Mladenović (2001).

O algoritmo VND, da Figura 32, pode ser visto como uma generalização do algoritmo de busca em vizinhança. A solução final proporcionada por esse algoritmo é um ótimo local em relação a todas as  $k_{max}$  estruturas de vizinhança e, portanto, a probabilidade de se alcançar um ótimo global é maior se comparado à aplicação de somente uma estrutura.

Logo, em cada passo, o algoritmo realiza buscas na vizinhança de um ponto que também é a incumbente do processo. Se nenhuma melhoria é alcançada em algum momento do processo, a vizinhança do ponto muda, de forma a considerar uma região maior do espaço de busca.

Segundo Hansen e Mladenović (2001), ao se implementar o algoritmo VND, as seguintes perguntas devem ser feitas:

- i. Qual a complexidade de diferentes tipos de movimentos?
- ii. Qual a melhor ordem para aplicá-los?
- iii. Os movimentos considerados são suficientes para assegurar a exploração completa da região contendo  $\mathbf{s}$  ?

iv. O quão precisa é a solução desejada?

A primeira pergunta refere-se à seleção e à classificação dos movimentos: se eles envolvem muitas mudanças elementares, a heurística resultante pode ser muito lenta e, geralmente, levar mais tempo do que um método exato para resolver um problema pequeno ou médio.

A segunda pergunta também tem influência no tempo computacional, em relação à qualidade das soluções obtidas. Uma implementação frequente consiste em realizar movimentos, por ordem crescente de complexidade, e voltar a realizar a busca considerando a primeira estrutura de vizinhança, cada vez que uma direção de descida é encontrada e um passo é dado naquela direção. Alternativamente, todos os movimentos podem ser aplicados em sequência, contanto que a descida seja feita em alguma das vizinhanças em sequência.

A terceira questão é crucial, pois, para alguns problemas, movimentos elementares são insuficientes para sair de uma região de vale, logo heurísticas que utilizam somente esses movimentos elementares conseguem obter soluções de baixa qualidade.

Finalmente, a precisão desejada (questão quatro) irá depender se o algoritmo VND é utilizado isoladamente ou em uma estrutura mais complexa, como a própria VNS. No primeiro caso, pode ser adequado realizar um esforço computacional grande para obter soluções de excelente qualidade. No segundo caso, é preferível obter uma solução de boa qualidade, rapidamente, por meio de um VND. Depois, melhorá-la, utilizando-se o algoritmo VNS Básico, o qual será descrito na seção 5.5.

#### 5.4 VNS REDUZIDA (RVNS)

Esse algoritmo foi inspirado em dois aspectos fundamentais no processo de busca, relacionados com a intensificação e a diversificação. O Fato 3 afirma que, na região de um ótimo local, normalmente, existem outras soluções ótimas locais que podem ser encontradas a partir de um ótimo local inicial e, portanto, deve-se montar uma estratégia de intensificação para tentar encontrar esses ótimos locais. Por outro lado, sair de um vale para encontrar um ótimo local de uma região mais distante exige uma estratégia que implique uma mudança mais radical na caracterização da vizinhança. Assim, uma busca que contemple ambos os aspectos (intensificação e diversificação) pode permitir encontrar ótimos locais de uma mesma região e pode permitir ao processo de busca sair para ótimos locais, de regiões mais distantes do ponto de ótimo local corrente (incumbente).

Considerando que se tenha encontrado um ótimo local  $s$  e que se pretenda, então,

encontrar outro de melhor qualidade. Nas versões básicas da VNS, assume-se que não há nenhum conhecimento prévio do espaço de busca. Portanto, segundo Hansen e Mladenović (2001), as perguntas a serem feitas são as seguintes:

- i. Em qual direção proceder a busca?
- ii. Qual a distância a ser percorrida?
- iii. Como modificar os movimentos se eles não produzem bons resultados?

A pergunta (i) está relacionada com a possibilidade de se alcançar qualquer ponto factível  $\mathbf{s} \in \mathcal{S}$ , ou todos os vales. A resposta mais simples para essa pergunta é: escolha uma direção aleatória. Para problemas com variáveis binárias (0 – 1), isto significa fazer o complemento de algumas variáveis. Para problemas euclidianos contínuos, considerar um coeficiente angular, ao acaso, faz com que todos os pontos de  $\mathcal{S}$  sejam considerados na busca.

A pergunta (ii) é crucial. Ao explorar o Fato 2, citado anteriormente até seus limites, ou seja, em muitos problemas combinatórios e de otimização global, ótimos locais tendem a estar próximos uns dos outros e estarem situados em uma região pequena de  $\mathcal{S}$ . Então, uma vez que um ótimo local tenha sido alcançado, ele contém informação implícita sobre ótimos locais, melhores e próximos, ou, até mesmo, de ótimos globais. É natural, então, explorar primeiramente as vizinhanças desse ótimo local. Todavia, se o vale que cerca esse ótimo local  $\mathbf{s}$  for grande, pode ser que isso não seja suficiente, e o que fazer em seguida é a resposta da questão (iii). Novamente, uma resposta natural é ir mais adiante.

Estes objetivos são almejados no algoritmo RVNS (do inglês *Reduced Variable Neighborhood Search*), apresentado na Figura 33. Um conjunto de estruturas de vizinhança  $\mathcal{N}_1(\mathbf{s}), \mathcal{N}_2(\mathbf{s}), \dots, \mathcal{N}_{k_{max}}(\mathbf{s})$  será considerado em torno do ponto corrente  $\mathbf{s}$  (que pode ou não ser um ótimo local). Geralmente, essas vizinhanças estarão aninhadas, ou seja, cada uma delas contém a anterior. Então, uma solução é escolhida aleatoriamente na primeira estrutura de vizinhança. Se esta solução for melhor que a incumbente (*i. e.*,  $v(\mathbf{s}') < v(\mathbf{s})$ ), a busca será recentralizada na nova solução ( $\mathbf{s} \leftarrow \mathbf{s}'$ ). Caso contrário, deve-se prosseguir para o próximo nível de vizinhança. Após todas as vizinhanças terem sido consideradas, a busca é reinicializada na primeira estrutura de vizinhança, até que o critério de parada seja satisfeito (geralmente este critério de parada é o tempo computacional máximo desde a última melhoria, ou um número máximo de iterações).

Devido à propriedade de aninhamento, o tamanho de estruturas de vizinhança sucessivas será crescente. Portanto, deve-se explorar mais minuciosamente as vizinhanças próximas de  $\mathbf{s}$  do que as distantes, e explorar as distantes somente quando nenhuma melhoria adicional for

observada nas primeiras, que são menores.

**Figura 33** - Algoritmo RVNS

**Inicialização:** defina as estruturas de vizinhança  $\mathcal{N}_k$ ,  $k = 1, \dots, k_{max}$ , que serão utilizadas na busca; encontre uma solução inicial  $\mathbf{s}$ ; defina um critério de parada;

**Repita** os passos seguintes até que o critério de parada esteja satisfeito:

(1) Faça  $k \leftarrow 1$ ;

(2) **Repita** os passos a seguir até  $k = k_{max}$ :

(a) **Agitação:** gere aleatoriamente uma solução  $\mathbf{s}'$  da  $k$  – ésima vizinhança de  $\mathbf{s}$  ( $\mathbf{s}' \in \mathcal{N}_k(\mathbf{s})$ );

(b) **Mover ou não:** se a solução  $\mathbf{s}'$  é melhor que a incumbente  $\mathbf{s}$ , mova para lá ( $\mathbf{s} \leftarrow \mathbf{s}'$ ) e continue a busca em  $\mathcal{N}_1$  ( $k \leftarrow 1$ ); caso contrário faça  $k \leftarrow k + 1$ .

Fonte: Hansen e Mladenović (2003a).

## 5.5 VNS BÁSICA (BVNS)

Algoritmos VNS mais eficientes podem ser formulados integrando-se as características do algoritmo VND, que permite encontrar ótimos locais de qualidade, e do algoritmo RVNS, que permite encontrar novas regiões promissoras a partir de um ótimo local. Assim, juntando-se essas características, podem ser formulados dois tipos de algoritmos VNS que, geralmente, apresentam excelente desempenho. Esses algoritmos são chamados de *Basic Variable Neighborhood Search* (BVNS) e *General Variable Neighborhood Search* (GVNS).

Mladenović e Hansen (1997) propuseram a primeira versão de um algoritmo VNS, o algoritmo VNS básico, para resolver o problema do caixeiro-viajante. Verificou-se que a metodologia proposta foi capaz de encontrar soluções de excelente qualidade para o problema. Denote-se por  $\mathcal{N}_k$ ,  $k = 1, \dots, k_{max}$  um conjunto finito de estruturas de vizinhanças pré-definidas, e por  $\mathcal{N}_k(\mathbf{s})$  o conjunto de soluções da  $k$  – ésima vizinhança de  $\mathbf{s}$ . Heurísticas de busca local geralmente utilizam apenas uma estrutura de vizinhança, ou seja,  $k_{max} = 1$ . Quando se utiliza mais de uma estrutura de vizinhança, as seguintes questões devem ser respondidas:

- i. Qual  $\mathcal{N}_k$  deve ser utilizado e quantas estruturas devem ser utilizadas?
- ii. Qual deve ser a ordem das estruturas de vizinhança na busca?
- iii. Qual estratégia deve ser utilizada para realizar as trocas de vizinhanças?

As respostas para essas perguntas são apresentadas em Mladenović e Hansen (1997),

por meio do algoritmo BVNS, apresentado na Figura 34.

**Figura 34** - Algoritmo BVNS

**Inicialização:** selecione o conjunto de estruturas de vizinhança  $\mathcal{N}_k$ ,  $k = 1, \dots, k_{max}$ , que será utilizado na busca; encontre uma solução inicial  $\mathbf{s}$ ; defina um critério de parada;

**Repita** os passos seguintes até que o critério de parada esteja satisfeito:

(1) Faça  $k \leftarrow 1$ ;

(2) Repita os passos a seguir até  $k = k_{max}$ :

(a) **Agitação:** gere aleatoriamente uma solução  $\mathbf{s}'$  da  $k$  – ésima vizinhança de  $\mathbf{s}$  ( $\mathbf{s}' \in \mathcal{N}_k(\mathbf{s})$ );

(b) **Busca local:** aplique algum método de busca local com  $\mathbf{s}'$  como solução inicial; denote por  $\mathbf{s}''$  o ótimo local obtido por essa busca;

(c) **Mover ou não:** se o ótimo local  $\mathbf{s}''$  melhor que a incumbente  $\mathbf{s}$ , mova para lá ( $\mathbf{s} \leftarrow \mathbf{s}''$ ) e continue a busca em  $\mathcal{N}_1$  ( $k \leftarrow 1$ ); caso contrário faça  $k \leftarrow k + 1$ .

Fonte: Mladenović e Hansen (1997).

O algoritmo BVNS combina a busca local com mudanças sistemáticas de estruturas de vizinhança em torno do ótimo local encontrado (HANSEN; MLADENOVIĆ, 2001).

De acordo com o esquema apresentado na Figura 34, inicialmente, a série de estruturas de vizinhança, as quais definem vizinhanças em torno do ponto  $\mathbf{s} \in \mathcal{S}$ , é selecionada. Depois, encontra-se uma solução inicial  $\mathbf{s}$ , podendo ser aplicada uma busca local para melhorá-la. O ponto  $\mathbf{s}'$  é selecionado aleatoriamente dentro da primeira estrutura de vizinhança  $\mathcal{N}_1(\mathbf{s})$  de  $\mathbf{s}$  e uma descida é realizada a partir de  $\mathbf{s}'$  utilizando a rotina de busca local. Isso leva a um novo ótimo local  $\mathbf{s}''$ . Nesse ponto, três resultados são possíveis:

- (i)  $\mathbf{s}'' = \mathbf{s}$ , isto é,  $\mathbf{s}$  já era ótimo local da vizinhança. Nesse caso, o procedimento continua na estrutura de vizinhança seguinte  $\mathcal{N}_k(\mathbf{s})$ ,  $k \geq 2$ ;
- (ii)  $\mathbf{s}'' \neq \mathbf{s}$  e  $v(\mathbf{s}'') \geq v(\mathbf{s})$ , ou seja, outro ótimo local foi encontrado, o qual não é melhor que a melhor solução anterior (ou incumbente). Nesse caso, o procedimento também vai para a próxima estrutura de vizinhança;
- (iii)  $\mathbf{s}'' \neq \mathbf{s}$  e  $v(\mathbf{s}'') \leq v(\mathbf{s})$ , ou seja, outro ótimo local, melhor que a incumbente foi encontrado; nesse caso, a busca é recentralizada em  $\mathbf{s}''$  e inicia novamente com a primeira vizinhança. Se a última vizinhança for alcançada sem que uma solução melhor que a incumbente tenha sido alcançada, a busca é reiniciada na

primeira estrutura de vizinhança  $\mathcal{N}_1(\mathbf{s})$  até que o critério de parada seja satisfeito.

O critério de parada do algoritmo BVNS pode ser, por exemplo, um número máximo de iterações, tempo de processamento máximo ou, ainda, máximo número de iterações entre duas melhorias. Geralmente, níveis de vizinhança sucessivos devem estar aninhados. Note que o ponto  $\mathbf{s}'$  é gerado aleatoriamente no passo 2(a), de forma a evitar ciclagem e permite encontrar ótimos locais distantes da incumbente corrente. Se a última vizinhança for alcançada sem que seja encontrada uma solução melhor que a incumbente, a busca é iniciada novamente na primeira vizinhança  $\mathcal{N}_1(\mathbf{s})$  até que uma condição de parada seja satisfeita.

## 5.6 VNS GERAL (GVNS)

A busca local no algoritmo BVNS pode ser qualquer estratégia heurística. Entretanto, também pode-se usar uma estratégia do algoritmo VNS. Assim, o algoritmo BVNS pode ser transformado em um algoritmo mais geral chamado *General Variable Neighborhood Search* (GVNS). O algoritmo GVNS é obtido generalizando o algoritmo BVNS simplesmente considerando um algoritmo VND como busca local e utilizando o algoritmo RVNS da Figura 33 para melhorar a solução inicial. A estrutura do algoritmo GVNS é mostrada na Figura 35.

**Figura 35** - Algoritmo GVNS

**Inicialização:** selecione um conjunto de estruturas de vizinhança  $\mathcal{N}_k$ ,  $k = 1, \dots, k_{max}$ , que será utilizado na etapa de agitação; selecione um conjunto de estruturas de vizinhança  $\mathcal{N}_l$ ,  $l = 1, \dots, l_{max}$ , que será utilizado na busca local; encontre uma solução inicial  $\mathbf{s}$  e melhore-a utilizando o algoritmo **RVNS**; defina um critério de parada;

**Repita** os passos seguintes até que o critério de parada esteja satisfeito:

(1) Faça  $k \leftarrow 1$ ;

(2) **Repita** os passos a seguir até  $k = k_{max}$ :

(a) **Agitação:** gere aleatoriamente uma solução  $\mathbf{s}'$  da  $k$  – ésima vizinhança de  $\mathbf{s}$  ( $\mathbf{s}' \in \mathcal{N}_k(\mathbf{s})$ );

(b) **Busca com VND:** aplique a busca **VND** com  $\mathbf{s}'$  como solução inicial e com as estruturas  $\mathcal{N}_l$ ,  $l = 1, \dots, l_{max}$ ; denote por  $\mathbf{s}''$  o ótimo local obtido por esta busca;

(c) **Mover ou não:** se a solução  $\mathbf{s}''$  é melhor que a incumbente  $\mathbf{s}$ , mova para lá ( $\mathbf{s} \leftarrow \mathbf{s}''$ ) e continue a busca em  $\mathcal{N}_1$  ( $k \leftarrow 1$ ); caso contrário faça  $k \leftarrow k + 1$ .



Diversas questões sobre como selecionar as estruturas de vizinhança são apresentadas por Hansen e Mladenović (2001) e podem ser conferidas a seguir:

- i. Quais propriedades das vizinhanças são obrigatórias para que o esquema resultante seja capaz de encontrar soluções ótimas globais ou quase ótimas?
- ii. Quais propriedades das vizinhanças serão úteis para encontrar soluções quase ótimas?
- iii. As estruturas de vizinhança devem estar aninhadas? Caso contrário, como elas devem estar ordenadas?
- iv. Quais são as propriedades desejáveis dos tamanhos das vizinhanças?

As duas primeiras questões tratam da habilidade da meta-heurística GVNS de encontrar os melhores vales e de fazer isso rapidamente. Para evitar ficar presa em um vale, quando podem existir vales mais profundos, a união das estruturas de vizinhança em torno de qualquer solução factível  $\mathbf{s}$  deve conter todo o conjunto de soluções factíveis, como mostrado em (5):

$$\mathbf{S} \subseteq \mathcal{N}_1(\mathbf{s}) \cup \mathcal{N}_2(\mathbf{s}) \cup \dots \cup \mathcal{N}_{k_{max}}(\mathbf{s}), \forall \mathbf{s} \in \mathbf{S} \quad (5)$$

Esses conjuntos devem cobrir  $\mathbf{S}$ , sem necessariamente particioná-lo, que é mais fácil de implementar, por exemplo, quando as vizinhanças são aninhadas, como mostrado em (6):

$$\mathcal{N}_1(\mathbf{s}) \subset \mathcal{N}_2(\mathbf{s}) \subset \dots \subset \mathcal{N}_{k_{max}}(\mathbf{s}), \mathbf{S} \subset \mathcal{N}_{k_{max}}(\mathbf{s}), \forall \mathbf{s} \in \mathbf{S} \quad (6)$$

Se essas propriedades não forem asseguradas, não se pode garantir que  $S$  possa ser completamente explorado.

Estruturas de vizinhança aninhadas são facilmente obtidas em muitos problemas combinatórios, definindo-se a primeira vizinhança  $\mathcal{N}_1(\mathbf{s})$  por um tipo de movimento e, em seguida, iterando-a  $k$  vezes para obter vizinhanças  $\mathcal{N}_k(\mathbf{s})$  para  $k = 1, \dots, k_{max}$ . Definidas dessa maneira, tais estruturas de vizinhança têm a propriedade de ter seus tamanhos (cardinalidade) crescentes. Portanto, se, como geralmente é o caso, percorre-se várias vezes a sequência de estruturas de vizinhança, as primeiras serão mais completamente exploradas do que as últimas. Isso é desejável tendo em vista o Fato 3, mencionado anteriormente, ou seja, os ótimos locais tendem a estar próximos uns dos outros.

Restringir movimentos ao conjunto de soluções factíveis  $\mathbf{S}$  pode ser muito restritivo, particularmente se o conjunto é não conexo. Uma possível solução para contornar esse problema é penalizar as infactibilidades na função objetivo.

## 5.7 ALGORITMO RVNS ESPECIALIZADO

A aplicação do algoritmo RVNS é preferida em relação aos outros algoritmos VNS quando se têm problemas em que a busca local é muito custosa. O problema de corte bidimensional não guilhotinado se enquadra nesse caso, pois para realizar a pesquisa em todo o espaço de busca seria necessário um grande esforço computacional, com relação ao tempo e ao armazenamento. Em virtude disso, a autora escolheu a meta-heurística RVNS como a segunda proposta a ser implementada para resolver o problema em estudo.

Para que o algoritmo RVNS torne-se especializado no problema de corte bidimensional não guilhotinado, a autora propôs quatro estruturas de vizinhança específicas e aplicáveis ao problema.

Deve-se observar que o algoritmo RVNS produz uma escolha de vizinhos mais dinâmica, escolhendo vizinhos de todas as estruturas de vizinhança (diversificação) e priorizando a primeira estrutura de vizinhança (intensificação) nas fases iniciais da busca. Entretanto, um componente importante da estrutura RVNS é a capacidade de encontrar novas regiões promissoras a partir de um ótimo local.

### 5.7.1 Funcionamento do Algoritmo RVNS Especializado

O algoritmo RVNS especializado começa sua execução gerando uma proposta solução inicial que é produzida pelo mesmo algoritmo heurístico construtivo proposto na seção 4.4, logo utiliza a mesma lógica aplicada na geração da população inicial do algoritmo genético de chaves aleatórias viciadas especializado para gerar a solução inicial do algoritmo RVNS especializado.

Após obter a solução inicial, são definidas as quatro estruturas de vizinhança que conduzirão a procura pela solução ótima no espaço de busca. Essas estruturas são definidas a seguir:

$\mathcal{N}_1$ : aleatoriamente, escolhe-se uma das peças que fazem parte da lista de peças cortadas na placa, dada pela solução corrente incumbente (padrão de corte obtido a partir do AHC), e se reduz, em uma fileira, o bloco composto por esse tipo de peça. Ou seja, acontecerá a diminuição da dimensão do bloco da peça escolhida, abrindo-se espaço para que uma possível peça que esteja fora do padrão de corte entre, ou que um bloco de uma peça que já esteja dentro do padrão de corte aumente.

$\mathcal{N}_2$ : aleatoriamente, são escolhidas duas peças que compõem o padrão de corte da proposta de solução corrente, a incumbente, e com essas duas peças aplica-se uma troca na ordem de corte. A troca entre elas pode proporcionar a mudança na ordem de todas as peças que são cortadas após a primeira peça trocada, portanto  $\mathcal{N}_2$  atua com uma vizinhança maior.

$\mathcal{N}_3$ : aleatoriamente, é escolhida uma peça que esteja fora do padrão de corte da proposta de solução corrente, a incumbente, e insere-se essa peça em uma posição, também aleatoriamente escolhida de dentro do padrão de corte, alterando-se toda a ordem de corte a partir da inserção da peça nova. A inserção da peça força o padrão de corte a sofrer uma reestruturação, portanto  $\mathcal{N}_3$  atua com uma vizinhança maior que  $\mathcal{N}_2$ .

$\mathcal{N}_4$ : aleatoriamente, são escolhidas três peças que compõem o padrão de corte da proposta de solução corrente, a incumbente, e com essas três peças aplica-se uma troca na ordem de corte. A troca entre elas força o padrão de corte a sofrer uma mudança ainda maior que a sofrida com  $\mathcal{N}_3$ , portanto  $\mathcal{N}_4$  atua com uma vizinhança maior.

A seguir, tem-se a descrição detalhada do funcionamento do algoritmo RVNS especializado.

- A. Gerar um vetor que traz uma representação de uma proposta de solução para o problema, onde essa representação é composta por duas parcelas, a primeira apresenta uma sequência das peças de 1 a  $m$ , gerada usando uma lógica (podendo ser aleatória), e a segunda parcela traz a sequência de orientação das peças. A orientação será fixa nesse primeiro momento.
- B. Decodificar o vetor. A primeira parcela do vetor ordenará as peças de maneira crescente. A segunda parcela contém a informação sobre a orientação de cada peça que consta na primeira parte do vetor.
- C. Utilizar o AHC para transformar o vetor ordenado em um padrão de corte. O AHC funciona seguindo os passos:
  - 1) Criar uma lista de retângulos livres existentes na placa, ou seja, todo os espaços que não foram utilizados para cortar uma peça dentro da placa são considerados livres e irão compor essa lista.
  - 2) Verificar qual é a primeira peça a ser cortada na placa, essa informação é obtida verificando-se a primeira parcela do vetor da proposta de solução.
  - 3) Criar um bloco retangular (ALVAREZ-VALDES; PARREÑO; TAMARIT, 2007) com peças iguais à primeira a ser cortada, respeitando seu limitante

superior. Já que cada peça não é necessariamente única, ela tem uma quantidade mínima e uma quantidade máxima de cópias que pode ser cortada na placa.

- 4) Observando-se que a origem da placa é seu canto inferior esquerdo e tendo sempre a origem como ponto de referência para o corte dos blocos de peças, cortar o bloco criado no passo 3, na origem da placa, seguindo a orientação da primeira peça. Essa informação é obtida verificando-se a segunda parcela do vetor da proposta de solução. Todos os blocos seguintes devem sempre ser cortados o mais próximo da origem (LAI; CHAN, 1997).
- 5) Após o bloco ser cortado na placa, atualizar a lista dos retângulos livres. Para obter os novos retângulos livres, a autora deste trabalho propõe um escaneamento da placa de cima para baixo e da esquerda para a direita. Após a obtenção dos retângulos livres, aplicar o processo de diferença e eliminação proposto por Lai e Chan (1997).
- 6) Verificar se há retângulos livres que possam ser utilizados pela próxima peça a ser cortada na placa, respeitando-se a ordem das peças, estabelecida pela primeira parcela do vetor da proposta de solução, e seguindo a orientação da peça, estabelecida pela segunda parcela do vetor da proposta de solução. Se não houver retângulo livre disponível para cortar a próxima peça respeitando a sua orientação, alterar a orientação da peça, girando-a, e verificar se, dessa maneira, existe algum retângulo livre que possa ser utilizado para o corte da peça. Se mesmo assim não existir retângulo livre disponível, essa peça não será cortada na placa e passa-se para a próxima peça, respeitando-se a ordem com que elas aparecem na primeira parcela do vetor da proposta de solução.
- 7) Havendo mais de um retângulo livre disponível para o corte da peça, escolher o retângulo que se encontra mais próximo da origem da placa (LAI; CHAN, 1997).
- 8) Formar um bloco retangular da peça (ALVAREZ-VALDES; PARREÑO; TAMARIT, 2007), respeitando-se as dimensões do retângulo livre escolhido e a quantidade de cópias da peça disponíveis para corte na placa.
- 9) Na placa, cortar o bloco formado no passo 8.
- 10) Voltar ao passo 5 e repetir a sequência de passos do 5 ao 9 até que nenhum retângulo livre possa ser cortado, usando-se, especificamente, as peças que ainda se encontram disponíveis para corte. Isso significa que mesmo depois de analisar todas as peças na ordem que aparecem no vetor e ainda existir algum retângulo

livre que possa ser cortado por uma peça que já foi cortada na placa, entretanto têm cópias dessa peça ainda disponíveis, essas cópias poderão ser utilizadas.

- D. Ao final do AHC, tem-se um padrão de corte composto por quatro vetores ou listas, a saber: o primeiro, com a ordem das peças cortadas na placa; o segundo, com a quantidade de cópias de cada peça cortada na placa; o terceiro, com a orientação de cada peça cortada na placa; o quarto, com as coordenadas de cada bloco cortado na placa.
- E. Após encontrar o padrão de corte, começa a estrutura do RVNS, composta por 4 estruturas de vizinhanças distintas.
- F. Entra-se em um ciclo de iterações, logo para-se somente quando for executado um número predeterminado de iterações.
- G. Dentro da estrutura de repetição, inicia-se a utilização da primeira estrutura de vizinhança  $\mathcal{N}_1$ .
- H. Aplicando-se  $\mathcal{N}_1$  um vizinho será encontrado. Se este vizinho for melhor do que a proposta corrente, o vizinho passa a ser a nova incumbente (proposta de solução corrente) e continua-se na estrutura de repetição aplicando  $\mathcal{N}_1$  até que o vizinho encontrado não seja melhor que a proposta de solução corrente, ou incumbente.
- I. Ao sair da estrutura de repetição onde se aplica  $\mathcal{N}_1$ , o algoritmo é levado a aplicar  $\mathcal{N}_2$ .
- J. Aplicando-se  $\mathcal{N}_2$  na incumbente, um vizinho será encontrado, se esse vizinho for melhor que a proposta de solução corrente incumbente, o vizinho passa a ser a nova proposta de solução incumbente e volta-se a aplicar a estrutura  $\mathcal{N}_1$  (passos de G à I), porém se o vizinho encontrado usando  $\mathcal{N}_2$  não for melhor do que a proposta de solução corrente incumbente, passa-se a aplicar a estrutura de vizinhança  $\mathcal{N}_3$  na incumbente.
- K. Aplicando-se  $\mathcal{N}_3$  na incumbente, um vizinho será encontrado. Se este vizinho for melhor que a proposta de solução incumbente, o vizinho passa a ser a proposta de solução corrente incumbente e volta-se a aplicar a estrutura  $\mathcal{N}_1$  (passos de G à I), porém se o vizinho encontrado usando  $\mathcal{N}_3$  não for melhor do que a proposta de solução corrente incumbente, passa-se a aplicar a estrutura de vizinhança  $\mathcal{N}_4$  na incumbente.
- L. Aplicando-se  $\mathcal{N}_4$  na incumbente, um vizinho será encontrado. Se este vizinho for melhor que a proposta de solução corrente incumbente, o vizinho passa a ser a proposta de solução corrente incumbente e volta-se a aplicar a estrutura  $\mathcal{N}_1$  (passos de G à I), porém se o vizinho encontrado usando  $\mathcal{N}_4$  não for melhor do que a proposta de solução corrente incumbente, mas sabendo que  $\mathcal{N}_4$  é a última estrutura de vizinhança aplicada nesta RVNS, o algoritmo, portanto volta a aplicar a estrutura  $\mathcal{N}_1$  (passos de G à I).

- M. Este processo se repete até que se tenha atingido um número predeterminado de iterações.
- N. Ao final do processo, tem-se o padrão de corte incumbente e seu valor de função objetivo.

## 6 RESULTADOS E DISCUSSÕES

Neste capítulo, o leitor encontrará a descrição dos testes realizados com as meta-heurísticas especializadas, propostas pela autora, para a resolução do problema de corte bidimensional não guilhotinado. Ainda neste capítulo, são apresentados os resultados dos testes realizados e, ao final, encontra-se a análise e discussão desses resultados.

### 6.1 TESTES

As duas meta-heurísticas especializadas em resolver o problema de corte bidimensional não guilhotinado, propostas por este trabalho, foram testadas utilizando-se casos de testes diferentes conhecidos na literatura.

Os casos de testes utilizados apresentam diferenças entre si nos seguintes aspectos: na quantidade de tipos de peças, nas dimensões delas e, até mesmo, nos números dos limitantes superiores e inferiores de cada uma. Estas instâncias de teste foram obtidas no site da OR-Library (2017). O conjunto de testes da OR-Library (2017) é composto de 21 casos extraídos da própria literatura especializada, sendo 12 instâncias de Beasley (1985), 2 instâncias de Hadjiconstantinou e Christofides (1995), 1 instância de Wang (1983), 1 instância de Christofides e Whitlock (1977) e 5 instâncias de Fekete e Scheppers (1997). Os dados dimensionais das peças e das placas usadas para os 21 casos de teste, bem como os limitantes inferiores e superiores utilizados em cada caso, podem ser encontrados no apêndice deste trabalho.

A importância de testar instâncias de pequeno, médio e grande porte, como as 21 instâncias citadas anteriormente, está no fato de poder verificar a performance dos algoritmos conforme aumenta a quantidade de tipos de peças e o tamanho da placa. Verificando se o desempenho dos algoritmos se mantém mesmo em instâncias maiores e, ainda, se a qualidade na solução encontrada é a mesma independentemente do tamanho do caso testado.

As meta-heurísticas foram executadas utilizando-se um notebook com processador Intel i7, 8GB de memória RAM, e utilizou-se também a linguagem de programação FORTRAN para o desenvolvimento delas. O tempo requerido para o processamento não se incrementou substancialmente nos diferentes casos de teste.

A título de comparação, os casos de teste também foram submetidos ao modelo matemático proposto por Hadjiconstantinou e Christofides (1995) para resolução do problema de corte bidimensional não guilhotinado com peças de orientação fixa. O modelo foi programado e testado no AMPL, utilizando como solver o CPLEX versão 1263. Não foi

encontrado, na literatura especializada, um modelo matemático para o problema em estudo onde fosse possível mudar a orientação das peças se necessário.

Na execução do AMPL, utilizou-se um servidor Dell | PowerEdge T430, com Sistema operacional Linux (distribuição Debian), com a seguinte configuração: 2x Processadores Intel® Xeon® E5-2650 v4 2.2GHz, 30M Cache, 9.60GT/s QPI, Turbo, HT, 12Cores/24Threads (105W) Max Mem 2400MHz, 64GB de memória, em 2x pentes de 32GB RDIMM, 2400MT/s, Dual Rank, x4 Data Width, 2x Discos Sólidos de 480GB SATA Read Intensive MLC 6Gbps 2.5” em carrier híbrido de 3.5” Hot Plug S3520.

Para realizar uma comparação justa com os resultados obtidos pelo modelo matemático testado no AMPL, os algoritmos especializados, propostos neste trabalho, tiveram que sofrer um pequeno ajuste. A orientação das peças dos 21 casos de teste foi considerada fixa, mantendo-se a equivalência nas restrições de resolução do problema em estudo.

Portanto, os 21 casos de teste foram testados com três diferentes algoritmos, são eles:

**Sistema I** - O modelo matemático, programado no AMPL, proposto por Hadjiconstantinou e Christofides (1995) para a resolução do problema de corte bidimensional não guilhotinado com orientação fixa de todas as peças em todos os testes;

**Sistema II** - Algoritmo genético de chaves aleatórias viciadas especializado com orientação fixa de todas as peças em todos os testes;

**Sistema III** - Algoritmo RVNS especializado com orientação fixa de todas as peças em todos os testes;

Consideraram-se estes testes comparativos importantes para se obter informações sobre a qualidade das soluções que as meta-heurísticas propostas pela autora conseguiram atingir, já que os resultados obtidos com a aplicação das metas-heurísticas foram comparados com os resultados obtidos por um modelo matemático equivalente.

Na literatura especializada, os testes também são realizados considerando a orientação fixa das peças. Geralmente, são usados os 21 casos de teste que foram utilizados neste trabalho, e, em alguns casos, adicionam-se aos testes instâncias reais fornecidas por indústrias de corte de matéria-prima. Na seção 6.6 deste trabalho, encontram-se os resultados mais relevantes na literatura especializada, obtidos pelos autores Beasley (1985) e Alvarez-Valdes, Parreño e Tamarit (2005), para comparação com os resultados encontrados pelos algoritmos que a autora propõe. Mais uma vez, os testes comparativos realizados, aqui, são considerados importantes, pois, dessa maneira, é possível verificar e comparar as melhorias nas soluções encontradas pelos autores que publicam na área e as melhorias apresentadas pelos algoritmos especializados



propostos pela autora em relação ao que já existe.

Um quarto algoritmo testou os 21 casos de teste, entretanto a orientação das peças de todos os casos de teste foi considerada livre, isto é, podendo variar a orientação, dependendo da necessidade do algoritmo em resolver o problema em estudo. O quarto algoritmo testado foi:

**Sistema IV** - Algoritmo genético de chaves aleatórias viciadas especializado com orientação livre de todas as peças em todos os testes;

A vantagem em se utilizar os algoritmos desenvolvidos neste trabalho, em relação ao modelo matemático, está no fato de que usando esses algoritmos não será preciso adquirir licença de software para utilizar programas de resolução de métodos exatos como é o caso do solver CPLEX em conjunto com o AMPL. Outra vantagem é que as meta-heurísticas desenvolvidas neste trabalho, utilizadas para a resolução do problema de corte bidimensional não guilhotinado, podem ser executadas em computadores domésticos que, geralmente, têm um valor menor em sua aquisição, principalmente quando comparados aos servidores que são necessários para se executar os métodos exatos. E outra vantagem em se utilizar as meta-heurísticas especializadas propostas neste trabalho, em relação aos modelos matemáticos, é que o algoritmo é próprio e com código-fonte conhecido, por isso é possível realizar modificações e customizações caso seja necessário.

## 6.2 RESULTADOS DOS TESTES COM O SISTEMA I

O modelo matemático proposto por Hadjiconstantinou e Christofides (1995), considerando as peças com orientação fixa, foi o primeiro sistema a ser testado. Utilizou-se do ambiente AMPL para programação, em conjunto com o solver CPLEX. Os resultados obtidos, após os 21 casos de teste, são exibidos na Tabela 6, onde encontram-se, na primeira parte da tabela, as informações sobre as instâncias de teste: número do caso de teste; dimensões da placa (Comprimento, Altura) e com quantos tipos de peça cada caso trabalha. Na segunda parte da Tabela 6, encontram-se os resultados obtidos em cada caso de teste pelo modelo matemático, contendo: o valor total da soma das peças cortadas na placa e a lista das peças usadas no corte da placa, todavia é preciso deixar claro que essa lista não representa a ordem de corte das peças na placa e, sim, quais foram as peças cortadas.

**Tabela 6** - Resultados Computacionais dos 21 Casos de Teste Usando o Modelo Matemático de Hadjiconstantinou e Christofides (1995), Programado no AMPL, Considerando a Orientação Fixa das Peças

Instâncias para Teste			Resultados Encontrados Sistema I	
Caso Teste	Placa (C, A)	Tipos de Peças	Total do Valor das Peças	Peças Usadas no Corte da Placa para Obtenção da Solução Ótima
1	(10, 10)	10	247	1, 1, 3, 3, 4, 8, 8
2	(10, 10)	7	230	1, 3, 3, 3, 4
3	(10, 10)	5	164	1, 2, 4, 4, 5
4	(15,10)	7	358	3, 3, 3, 4, 6, 7
5	(15,10)	5	268	1, 2, 3, 4, 5, 5
6	(15, 10)	10	289	1, 5, 6, 7, 8, 9, 10
7	(20, 20)	7	834	1, 2, 2, 2, 3, 4, 5, 6
8	(20, 20)	5	430	1, 1, 1, 2, 3, 4, 5, 5
9	(20, 20)	10	924	1, 2, 3, 4, 4, 4, 7, 8, 10, 10, 10
10	(30, 30)	7	1688	1, 1, 1, 3, 3, 4, 5, 5, 7
11	(30, 30)	7	1178	1, 2, 3, 6, 7
12	(30, 30)	10	1865	1, 3, 5, 5, 7, 8, 8, 9, 10
13	(30, 30)	5	1452	1, 1, 1, 3, 3, 3
14	(30, 30)	15	1270	1, 5, 8, 13, 14
15	(70, 40)	19	2726	2, 2, 2, 3, 3, 5, 5, 13
16	(40, 70)	20	1860	1, 2, 3, 13, 20
17	(100, 100)	15	27718	3, 5, 5, 6, 6, 9, 9, 9, 12, 13, 13
18	(100, 100)	30	22502	4, 6, 7, 9, 10, 11, 14, 15, 18, 24, 27
19	(100, 100)	30	24019	6, 9, 12, 13, 16, 17, 19, 24, 25, 27, 29
20	(100, 100)	33	32893	8, 9, 9, 21, 22, 22, 22, 32, 32, 33
21	(100, 100)	29	27923	7, 7, 7, 7, 7, 14, 14, 14

Fonte: Elaborado pela autora.

Os 15 primeiros casos tiveram tempo de execução de 1 a 10 segundos, porém os 6 últimos casos tiveram tempo de execução superior a 20 horas. A seguir tem-se a descrição detalhada de cada caso.

Caso 1: Tempo de execução de 1 segundo.

Caso 2: Tempo de execução de 1 segundo.

Caso 3: Tempo de execução de 1 segundo.

- Caso 4: Tempo de execução de 2 segundo.  
 Caso 5: Tempo de execução de 2 segundo.  
 Caso 6: Tempo de execução de 3 segundo.  
 Caso 7: Tempo de execução de 3 segundo.  
 Caso 8: Tempo de execução de 3 segundo.  
 Caso 9: Tempo de execução de 5 segundo.  
 Caso 10: Tempo de execução de 5 segundo.  
 Caso 11: Tempo de execução de 5 segundo.  
 Caso 12: Tempo de execução de 7 segundo.  
 Caso 13: Tempo de execução de 4 segundo.  
 Caso 14: Tempo de execução de 9 segundo.  
 Caso 15: Tempo de execução de 10 segundo.  
 Caso 16: Tempo de execução de 20 horas.  
 Caso 17: Tempo de execução de 29 horas.  
 Caso 18: Tempo de execução de 32 horas.  
 Caso 19: Tempo de execução de 32 horas.  
 Caso 20: Tempo de execução de 35 horas.  
 Caso 21: Tempo de execução de 30 horas.

### 6.3 RESULTADOS DOS TESTES COM O SISTEMA II

O algoritmo genético de chaves aleatórias viciadas especializado, considerando fixa a orientação das peças, foi o segundo a ser testado. Os testes foram repetidos vinte vezes para os 21 casos de teste. Ao final, foi possível calcular a média dos resultados encontrados e verificar o melhor resultado para cada caso de teste. A Tabela 7, a seguir, traz uma análise dos resultados encontrados durante a execução de dois casos de teste, caso 2 e 21, para exemplificar o trabalho desenvolvido durante os testes.

**Tabela 7** - Análise dos Resultados Encontrados Durante os Vinte Testes Realizados – Sistema II

Análise dos Resultados Encontrados			
Caso Teste	Menor Valor	Melhor Valor	Média dos 20 Testes
2	228	230	229,55
21	26192	27923	27490,25

Fonte: Elaborado pela autora.

Na Tabela 8, mostram-se os melhores resultados obtidos nos 21 casos de teste utilizando esse algoritmo. Na primeira parte da Tabela 8, encontram-se as especificações dos casos de teste, contendo: o número do teste; as dimensões da placa e com quantos tipos de peças cada caso trabalha. Na segunda parte da Tabela 8, encontram-se os resultados obtidos pelo algoritmo especializado em cada caso de teste, contendo: o valor total da soma da peças cortadas na placa; a lista de ordem de corte das peças; a quantidade de exemplares das peças contidas na lista de ordem de corte formando um bloco e, por último, as coordenadas dos blocos cortados na placa.

Utilizou-se, nos algoritmos especializados desenvolvidos pela autora, a representação de proposta de solução também sugerida por ela, conforme seção 3.5. Por esse motivo, as tabelas que apresentam os resultados dos testes encontrados pelos algoritmos especializados trazem como padrão de corte várias informações, diferenciando-se do modelo matemático que somente representa a solução do problema com as peças utilizadas na solução e o valor total das peças cortadas na placa.

**Tabela 8** - Resultados Computacionais dos 21 Casos de Teste Usando o Algoritmo Genético de Chaves Aleatórias Viciadas Especializado Considerando a Orientação Fixa das Peças

Instâncias para Teste			Resultados Encontrados (Padrão de Corte)			
Caso Teste	Placa (C, A)	Tipos de Peças	Sistema II			
			Total Valor Peças	Ordem de Corte das Peças	Qtd. De Peças que Formam os Blocos	Coordenadas dos Blocos
1	(10, 10)	10	234	5, 3, 6, 10	1, 2, 1, 1	BLOCO 5(0,0)(9,10) BLOCO 3(2,0)(9,5) BLOCO 6 (2,6)(9,9) BLOCO 10 (2,10)(10,10)
2	(10, 10)	7	230	3, 1, 4	3, 1, 1	BLOCO 3(0,0)(9,9) BLOCO 1(10,0)(10,10) BLOCO 4(1,10)(6,10)
3	(10, 10)	5	156	5, 1	2, 2	BLOCO 5(0,0)(4,9) BLOCO 1(5,0)(10,7)
4	(15,10)	7	358	4, 3, 6, 7	1, 3, 1, 1	BLOCO 4(0,0)(11,3) BLOCO 3(0,4)(13,9) BLOCO 6(0,10)(11,10) BLOCO 7(14,0)(15,10)
5	(15,10)	5	268	3, 2, 5, 1, 4	1, 1, 2, 1, 1	BLOCO 3(0,0)(0,9) BLOCO 2(0,2)(7,4) BLOCO 5(0,5)(12,8) BLOCO 1(0,9)(15,10) BLOCO 4(8,2)(15,4)
6	(15, 10)	10	289	1, 9, 5, 8, 10, 7, 6	1, 1, 1, 1, 1, 1, 1	BLOCO 1(0,0)(2,9) BLOCO 9(3,0)(13,2) BLOCO 5(3,3)(5,10) BLOCO 8(6,3)(15,5)

						BLOCO 10(6,6)(9,10) BLOCO 7(10,6)(15,9) BLOCO 6(10,10)(13,10)
7	(20, 20)	7	828	2, 3, 4, 5	3, 2, 2, 1	BLOCO 2(0,0)(18,6) BLOCO 3(0,7)(12,8) BLOCO 4(0,9)(16,15) BLOCO 5 (0,16)(18,20)
8	(20, 20)	5	430	5, 2, 4, 3, 1	2, 1, 1, 1, 3	BLOCO 5 (0,0)(3,2) BLOCO 2(0,3)(16,5) BLOCO 4(0,6)(20,7) BLOCO 3(0,8)(18,10) BLOCO 1(0,11)(3,19)
9	(20, 20)	10	924	7, 8, 1, 3, 4, 2, 10	1, 1, 1, 1, 3, 1, 3	BLOCO 7 (0,0)(7,9) BLOCO 8(0,10)(17,14) BLOCO 1(0,15)(14,16) BLOCO 3(0,17)(20,20) BLOCO 4(8,0)(19,9) BLOCO 2(20,0)(20,5) BLOCO 10(18,10)(20,16)
10	(30, 30)	7	1688	5, 1, 7, 4, 3	2, 3, 1, 1, 2	BLOCO 5(0,0)(28,7) BLOCO 1(0,8)(9,3) BLOCO 7(10,8)(30,15) BLOCO 4(10,16)(26,26) BLOCO 3(10,27)(30,30)
11	(30, 30)	7	1178	6, 2, 3, 1, 7	1, 1, 1, 1, 1	BLOCO 6(0,0)(16,6) BLOCO 2(0,7)(17,27) BLOCO 3(17,0)(22,6) BLOCO 1(18,7)(29,27) BLOCO 7(23,0)(27,4)
12	(30, 30)	10	1865	9, 1, 10, 8, 3, 7, 5	1, 1, 1, 2, 1, 1, 2	BLOCO 9(0,0)(4,30) BLOCO 1(5,0)(20,19) BLOCO 10(5,7)(22,30) BLOCO 8(5,20)(22,30) BLOCO 3(23,0)(28,28) BLOCO 7(29,0)(30,30) BLOCO 5(23,29)(27,30)
13	(30, 30)	5	1452	3, 1	3, 3	BLOCO 3(0,9)(27,27) BLOCO 1(28,0)(30,30)
14	(30, 30)	15	1270	1, 5, 8, 13, 14	1, 1, 1, 1, 1	BLOCO 1(0,0)(22,21) BLOCO 5(23,0)(6,27) BLOCO 8(0,22)(6,27) BLOCO 13(7,22)(11,27) BLOCO 14(12,22)(27,27)
15	(70, 40)	19	2667	7, 4, 3, 2	2, 4, 1, 1	BLOCO 7(0,0)(54,17) BLOCO 4(0,18)(56,40) BLOCO 3(57,0)(68,21) BLOCO 2(57,22)(67,40)
16	(40, 70)	20	1820	18, 19, 15, 20, 14	3, 4, 1, 1, 1	BLOCO 18(0,0)(21,36) BLOCO 19(22,0)(40,44) BLOCO 15(0,37)(16,61) BLOCO 20(17,45)(25,61) BLOCO 14(26,45)(40,68)
17	(100, 100)	15	27539	9, 3, 6, 12, 5, 13	3, 1, 2, 1, 2, 1	BLOCO 9(0,0)(70,81) BLOCO 3(71,0)(92,80) BLOCO 6(0,82)(100,99) BLOCO 12(0,100)(100,100) BLOCO 5(93,0)(98,80)

						BLOCO 13(99,0)(100,41)
18	(100, 100)	30	22502	4, 24, 6, 18, 7, 14, 15, 11, 10, 27, 9	1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1	BLOCO 4(0,0)(6,80) BLOCO 24(7,0)(83,31) BLOCO 6(84,0)(89,86) BLOCO 18(90,0)(100,84) BLOCO 7(7,32)(64,51) BLOCO 14(7,52)(48,83) BLOCO 15(49,52)(65,81) BLOCO 11(66,32)(72,84) BLOCO 10(1,87)(100,100) BLOCO 27(65,32)(65,37) BLOCO 9(73,32)(81,83)
19	(100, 100)	30	24019	16, 13, 17, 19, 25, 9, 27, 12, 29, 6, 24	1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1	BLOCO 16(0,0)(28,82) BLOCO 13(29,0)(47,68) BLOCO 17(48,0)(59,65) BLOCO 19(60,0)(80,96) BLOCO 25(29,69)(44,89) BLOCO 9(81,0)(90,97) BLOCO 27(91,0)(95,89) BLOCO 12(45,69)(57,80) BLOCO 29(0,90)(41,95) BLOCO 6(0,98)(100,100) BLOCO 24(96,0)(100,84)
20	(100, 100)	33	32893	9, 21, 32, 33, 22, 8	2, 1, 2, 1, 3, 1	BLOCO 9(0,0)(66,72) BLOCO 21(0,73)(62,100) BLOCO 32(67,0)(92,99) BLOCO 33(93,0)(99,76) BLOCO 22(93,77)(98,97) BLOCO 8(100,0)(100,97)
21	(100, 100)	29	27923	7, 14	5, 3	BLOCO 7(0,0)(58,100) BLOCO 14(59,0)(100,96)

Fonte: Elaborado pela autora.

O tempo requerido para o processamento não se incrementou substancialmente nos diferentes casos de teste, atingindo um tempo de execução, em média, de 5 segundos.

#### 6.4 RESULTADOS DOS TESTES COM O SISTEMA III

O algoritmo RVNS especializado para o problema de corte bidimensional não guilhotinado, considerando fixa a orientação das peças, foi o terceiro algoritmo testado. Os testes com esse sistema também foram repetidos vinte vezes para cada caso de teste. Usando-se a mesma disposição de informações que a Tabela 8, a Tabela 9 mostra os melhores resultados obtidos nos 21 casos de teste, utilizando-se esse algoritmo.

**Tabela 9** - Resultados Computacionais dos 21 Casos de Teste Usando o Algoritmo RVNS Especializado Considerando a Orientação Fixa das Peças

Instâncias para Teste	Resultados Encontrados (Padrão de Corte) Sistema III
--------------------------	---

Caso Teste	Placa (C, A)	Tipos de Peças	Total Valor Peças	Ordem de Corte das Peças	Qtd. De Peças que Formam os Blocos	Coordenadas dos Blocos
1	(10, 10)	10	230	8, 10, 5, 4	3, 2, 1, 2	BLOCO 8(0,0)(9,7) BLOCO 10(0,8)(9,9) BLOCO 5(10,0)(10,10) BLOCO 4(0,10)(8,10)
2	(10, 10)	7	228	1, 6, 2, 4, 1	3, 1, 2, 2, 1	BLOCO 1(0,0)(3,10) BLOCO 6(4,0)(7,0) BLOCO 2(4,2)(8,7) BLOCO 4(9,0)(9,6) BLOCO 1(10,0)(10,10)
3	(10, 10)	5	156	5, 1	2, 2	BLOCO 5(0,0)(4,9) BLOCO 1(5,0)(10,7)
4	(15,10)	7	358	4, 3, 6, 7	1, 3, 1, 1	BLOCO 4(0,0)(11,3) BLOCO 3(0,4)(13,9) BLOCO 6(0,10)(11,10) BLOCO 7(14,0)(15,10)
5	(15,10)	5	268	3, 2, 5, 1, 4	1, 1, 2, 1, 1	BLOCO 3(0,0)(0,9) BLOCO 2(0,2)(7,4) BLOCO 5(0,5)(12,8) BLOCO 1(0,9)(15,10) BLOCO 4(8,2)(15,4)
6	(15, 10)	10	289	1, 9, 5, 8, 10, 7, 6	1, 1, 1, 1, 1, 1, 1	BLOCO 1(0,0)(2,9) BLOCO 9(3,0)(13,2) BLOCO 5(3,3)(5,10) BLOCO 8(6,3)(15,5) BLOCO 10(6,6)(9,10) BLOCO 7(10,6)(15,9) BLOCO 6(10,10)(13,10)
7	(20, 20)	7	805	7, 5, 2, 3, 2	1, 2, 3, 1, 1	BLOCO 7(0,0)(8,3) BLOCO 5(0,4)(18,13) BLOCO 2(0,14)(12,19) BLOCO 3(0,20)(13,20) BLOCO 2(13,14)(19,19)
8	(20, 20)	5	430	5, 2, 4, 3, 1	2, 1, 1, 1, 3	BLOCO 5(0,0)(6,0) BLOCO 2(0,2)(16,4) BLOCO 4(0,5)(20,6) BLOCO 3(0,7)(18,9) BLOCO 1(0,10)(2,18)
9	(20, 20)	10	924	7, 8, 1, 3, 4, 2, 10	1, 1, 1, 1, 3, 1, 3	BLOCO 7(0,0)(7,9) BLOCO 8(0,10)(17,14) BLOCO 1(0,15)(14,16) BLOCO 3(0,17)(20,20) BLOCO 4(8,0)(19,9) BLOCO 2(20,0)(20,5) BLOCO 10(18,10)(20,16)
10	(30, 30)	7	1686	5, 4, 6, 1, 3	2, 2, 2, 1, 2	BLOCO 5(0,0)(28,7) BLOCO 4(71,17)(92,80) BLOCO 6(0,82)(100,99) BLOCO 1(0,100)(100,100) BLOCO 3(93,0)(98,80)
11	(30, 30)	7	1178	6, 2, 3, 1, 7	1, 1, 1, 1, 1	BLOCO 6(0,0)(16,6) BLOCO 2(0,7)(17,27) BLOCO 3(17,0)(22,6)

						BLOCO 1(18,7)(29,27) BLOCO 7(23,0)(27,4)
12	(30, 30)	10	1829	6, 7, 5, 3, 7, 6	2, 2, 2, 2, 1, 1	BLOCO 6(0,0)(12,26) BLOCO 7(13,0)(16,3) BLOCO 5(17,0)(26,0) BLOCO 3(17,2)(22,24) BLOCO 7(29,0)(30,30) BLOCO 6(23,2)(28,27)
13	(30, 30)	5	1452	3, 1	3, 3	BLOCO 3(0,9)(27,27) BLOCO 1(28,0)(30,30)
14	(30, 30)	15	1270	1, 5, 8, 13, 14	1, 1, 1, 1, 1	BLOCO 1(0,0)(22,21) BLOCO 5(23,0)(6,27) BLOCO 8(0,22)(6,27) BLOCO 13(7,22)(11,27) BLOCO 14(12,22)(27,27)
15	(70, 40)	19	2505	12, 1, 5	2, 1, 2	BLOCO 12(0,0)(64,24) BLOCO 1(25,0)(17,33) BLOCO 5(19,25)(65,11)
16	(40, 70)	20	1820	18, 19, 15, 20, 14	3, 4, 1, 1, 1	BLOCO 18(0,0)(21,36) BLOCO 19(22,0)(40,44) BLOCO 15(0,37)(16,61) BLOCO 20(17,45)(25,61) BLOCO 14(26,45)(40,68)
17	(100, 100)	15	27517	6, 12, 5, 15, 13, 6	4, 5, 5, 2, 5, 1	BLOCO 6(0,0)(100, 36) BLOCO 12(0,37)(100,41) BLOCO 5(0,42)(30,81) BLOCO 15(31,60)(81,89) BLOCO 13(82,42)(91,82) BLOCO 6(0,90)(100,98)
18	(100, 100)	30	21508	28, 27, 7, 29, 17, 15, 25, 3, 1	1, 1, 1, 1, 1, 1, 1, 1, 1	BLOCO 28(0,0)(12,99) BLOCO 27(13,0)(13,6) BLOCO 7(14,0)(71,20) BLOCO 29(13,21)(45,92) BLOCO 17(72,0)(97,65) BLOCO 15(46,21)(62,50) BLOCO 25(98,0)(100,71) BLOCO 3(46,66)(87,84) BLOCO 1(13,93)(93,100)
19	(100, 100)	30	23943	17, 10, 8, 11, 29, 7	1, 1, 1, 1, 1, 1	BLOCO 17(0,0)(12,65) BLOCO 10(13,0)(62,47) BLOCO 8(63,0)(85,95) BLOCO 11(13,48)(53,92) BLOCO 29(13,93)(53,98) BLOCO 7(86,0)(95,9)
20	(100, 100)	33	30067	29, 11, 22, 22, 7	2, 3, 2, 1, 2	BLOCO 29(0,0)(46,80) BLOCO 11(57,0)(85,80) BLOCO 22(0,81)(12,87) BLOCO 22(13,81)(18,87) BLOCO 7(86,0)(95,48)
21	(100, 100)	29	27790	7, 27, 29, 14, 8, 8	4, 5, 4, 2, 2, 2	BLOCO 7(0,0)(58,20) BLOCO 27(59,0)(83,6) BLOCO 29(0,21)(84,46) BLOCO 14(0,47)(84,78) BLOCO 8(85,0)(90,99) BLOCO 8(91,0)(96,99)

Fonte: Elaborado pela autora.



O mesmo acontece com o Sistema III, em relação ao tempo de processamento, se comparado ao Sistema II. Ele não se alterou diante dos diferentes casos de teste, atingindo um tempo de execução em média de 5 segundos.

## 6.5 RESULTADOS DOS TESTES COM O SISTEMA IV

O algoritmo genético de chaves aleatórias viciadas especializado, considerando as peças com orientação livre, foi o último algoritmo testado. Os testes foram repetidos vinte vezes para os 21 casos de teste. Ao final, foi possível calcular a média dos resultados encontrados e verificar o melhor deles para cada caso de teste. A Tabela 10, a seguir, traz uma análise dos resultados encontrados durante a execução de dois casos de teste, caso 2 e 21, os mesmos também utilizados como exemplo para o Sistemas II.

**Tabela 10** - Análise dos Resultados Encontrados Durante os Vinte Testes Realizados – Sistema IV

Análise dos Resultados Encontrados			
Caso Teste	Menor Valor	Melhor Valor	Média dos 20 Testes
2	244	250	248,95
21	26332	27983	27640,80

Fonte: Elaborado pela autora.

Na Tabela 11, mostram-se os melhores resultados obtidos pelo algoritmo nos 21 casos de teste. A disposição das informações, na 11, é semelhante às usadas nas Tabelas 8 e 9, o que difere é a coluna de orientação das peças/bloco, onde, nessa coluna, obtém-se a informação sobre qual orientação a peça/bloco (todas as peças que compõem um bloco têm a mesma orientação) assumiu ao ser cortada na placa.

**Tabela 11** - Resultados Computacionais dos 21 Casos de Teste com o Algoritmo Genético de Chaves Aleatórias Viciadas Especializado Considerando a Orientação Livre das Peças

Instâncias para Teste			Resultados Encontrados (Padrão de Corte)				
			Sistema IV				
Caso Teste	Placa (C, A)	Tipos de Peças	Total Valor Peças	Ordem de Corte das Peças	Qtd. De Peças que Formam os Blocos	Orientação das peças/bloco	Coordenadas dos Blocos
1	(10, 10)	10	259	3, 4, 10, 6	2, 2, 2, 1	1, 1, 0, 0	BLOCO 3(0,0)(0,10)

							BLOCO 4(0,5)(2,8) BLOCO 10(0,9)(9,10) BLOCO 6(3,5)(10,8)
2	(10, 10)	7	250	1, 2, 3, 4, 6	3, 2, 1, 1, 1	1, 0, 0, 0, 1	BLOCO 1(0,0)(10,3) BLOCO 2(0,4)(10,6) BLOCO 3(0,7)(9,9) BLOCO 4(0,10)(6,10) BLOCO 6(10,7)(10,10)
3	(10, 10)	5	193	3, 5, 2	1, 2, 2	1, 0, 1	BLOCO 3(0,0)(2,10) BLOCO 5(3,0)(6,9) BLOCO 2(7,0)(10,8)
4	(15,10)	7	370	4, 3, 2, 6	1, 3, 1, 1	0, 0, 1, 0	BLOCO 4(0,0)(11,3) BLOCO 3(0,4)(12,9) BLOCO 2(13,0)(15,9) BLOCO 6(0,10)(11,10)
5	(15,10)	5	268	3, 4, 5, 1, 2	1, 1, 2, 1, 1	0, 0, 0, 0, 0	BLOCO 3(0,0)(9,0) BLOCO 4(0,2)(8,4) BLOCO 5(0,5)(12,8) BLOCO 1(0,9)(15,10) BLOCO 2(9,2)(15,4)
6	(15, 10)	10	298	8, 9, 5, 6, 3, 1, 4, 10	1, 1, 1, 1, 1, 1, 1, 1	1, 0, 0, 1, 0, 1, 1, 1	BLOCO 8(0,0)(3,10) BLOCO 9(4,0)(14,2) BLOCO 5(4,3)(6,10) BLOCO 6(15,0)(15,4) BLOCO 3(7,3)(8,6) BLOCO 1(7,7)(15,8) BLOCO 10(9,3)(13,6)
7	(20, 20)	7	856	1, 2, 3, 4, 6, 7	1, 3, 2, 2, 1, 1	0, 1, 1, 0, 0, 0	BLOCO 1(0,0)(15,3) BLOCO 2(0,4)(18,9) BLOCO 3(19,0)(20,13) BLOCO 4(0,10)(16,16) BLOCO 6(0,17)(12,20) BLOCO 7(13,17)(20,19)
8	(20, 20)	5	430	1, 4, 2, 3, 5	3, 1, 1, 1, 2	0, 0, 0, 0, 1	BLOCO 1(0,0)(3,9) BLOCO 4(0,10)(20,11) BLOCO 2(0,12)(16,14) BLOCO 3(0,15)(18,17) BLOCO 5(4,3)(5,3)
9	(20, 20)	10	924	1, 8, 4, 3, 7, 2, 10, 10,	1, 1, 3, 1, 1, 1, 2, 1	0, 0, 0, 0, 0, 0, 0, 0	BLOCO 1(0,0)(20,20) BLOCO 8(0,3)(17,7) BLOCO 4(0,8)(12,16) BLOCO 3(0,17)(20,20) BLOCO 7(13,8)(19,16) BLOCO 2(18,0)(18,5) BLOCO 10(19,0)(20,7) BLOCO 10(20,8)(20,14)
10	(30, 30)	7	1786	3, 4, 5, 7	3, 2, 2, 1	0, 1, 0, 1	BLOCO 3(0,0)(21,6) BLOCO 4(0,7)(22,23) BLOCO 5(0,24)(14,30) BLOCO 7(23,0)(30,21)
11	(30, 30)	7	1272	1, 4, 2, 7, 5	1, 1, 1, 1, 1	1, 0, 1, 1, 0	BLOCO 1(0,0)(21,12) BLOCO 4(22,0)(30,15) BLOCO 2(0,13)(21,29) BLOCO 7(22,16)(25,20) BLOCO 5(26,17)(30,27)

12	(30, 30)	10	1932	3, 7, 8, 5, 2	3, 1, 3, 2, 1	1, 0, 0, 1, 0	BLOCO 3(0,0)(28,18) BLOCO 7(29,0)(30,30) BLOCO 8(0,19)(27,29) BLOCO 5(28,19)(28,28) BLOCO 2(0,30)(24,30)
13	(30, 30)	5	1452	1, 3	3, 3	1, 1	BLOCO 1(0,0)(3,30) BLOCO 3(4,0)(30,27)
14	(30, 30)	15	1431	1, 6, 5, 8, 13, 15	1, 1, 1, 1, 1, 1	1, 0, 1, 0, 0, 1	BLOCO 1(0,0)(21,22) BLOCO 6(22,0)(30,17) BLOCO 5(0,23)(21,30) BLOCO 8(22,18)(27,23) BLOCO 13(22,24)(26,29) BLOCO 15(27,24)(30,28)
15	(70, 40)	19	2793	19, 15,1	1, 1, 1	0, 1, 0	BLOCO 19(0,0)(43,31) BLOCO 15(44,0)(70,37) BLOCO 1(0,32)(17,40)
16	(40, 70)	20	1840	15, 19, 18, 20	2, 4, 3, 1	0, 0, 0, 1	BLOCO 15(0,0)(32,25) BLOCO 19(0,26)(19,69) BLOCO 18(20,26)(40,61) BLOCO 20(20,62)(36,70)
17	(100, 100)	15	28090	6, 15, 1, 13	5, 4, 1, 2	1, 0, 0, 1	BLOCO 6(0,0)(41,100) BLOCO 15(46,0)(96,96) BLOCO 1(97,0)(100,90) BLOCO 13(46,97)(86,100)
18	(100, 100)	30	22852	18, 8, 23, 17, 6, 14, 7, 15, 30, 27, 11, 9	1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1	1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0	BLOCO 18(0,0)(84,11) BLOCO 8(0,12)(99,14) BLOCO 23(0,15)(93,31) BLOCO 17(32,96)(26,96) BLOCO 6(94,15)(99,100) BLOCO 14(27,32)(58,73) BLOCO 7(27,74)(84,93) BLOCO 15(59,32)(88,48) BLOCO 30(59,84)(49,69) BLOCO 27(89,32)(89,37) BLOCO 11(27,94)(79,100) BLOCO 9(85,46)(93,80)
19	(100, 100)	30	24811	9, 16, 5, 13, 12, 30, 17, 29, 1, 27	1, 1, 1, 1, 1, 1, 1, 1, 1, 1	0, 0, 1, 0, 0, 1, 0, 0, 1, 1	BLOCO 9(0,0)(10,97) BLOCO 16(11,0)(38,32) BLOCO 5(11,33)(24,88) BLOCO 13(39,0)(57,68) BLOCO 12(45,69)(57,80) BLOCO 30(58,0)(87,76) BLOCO 17(88,0)(99,65) BLOCO 29(58,77)(98,82) BLOCO 1(0,98)(98,100) BLOCO 27(11,89)(98,93)
20	(100, 100)	33	32893	9, 32, 33, 21, 8, 22	2, 2, 1, 1, 1, 3	0, 0, 0, 0, 0, 1	BLOCO 9(0,0)(66,72) BLOCO 32(67,0)(92,99) BLOCO 33(93,0)(99,76) BLOCO 21(0,73)(28,100) BLOCO 8(100,0)(100,97) BLOCO 29(93,77)(99,82)
21	(100, 100)	29	27983	7, 14, 27	5, 3, 5	1, 1, 0	BLOCO 7(0,0)(100,58) BLOCO 14(0,59)(96,100) BLOCO 27(97,59)(97,93)

Fonte: Elaborado pela autora.

O mesmo acontece com o Sistema IV em relação ao tempo de processamento se comparado ao Sistema II e III. Ele não se alterou diante dos diferentes casos de teste, atingindo um tempo de execução, em média, de 5 segundos.

## 6.6 ANÁLISE E DISCUSSÕES SOBRE OS RESULTADOS

Os algoritmos especializados, propostos neste trabalho, foram capazes de resolver o problema de corte bidimensional não guilhotinado. Os resultados encontrados foram exibidos nas Tabelas 8, 9, e 11, contudo, para que se pudesse mensurar a qualidade desses resultados, algumas comparações foram realizadas.

Na Tabela 12, é possível analisar os resultados obtidos por este trabalho, comparando-os com os resultados de trabalhos já publicados e tidos como referência na literatura especializada.

**Tabela 12** - Comparação de Resultados Computacionais Considerando Orientação Fixa

Comparação dos Valores Totais das Peças Cortadas com Orientação Fixa					
Caso de Teste	Beasley <sup>1</sup>	GRASP <sup>2</sup>	Sistema I	Sistema II	Sistema III
1	247	247	247	234	230
2	230	230	230	230	228
3	164	164	164	156	156
4	358	358	358	358	358
5	268	268	268	268	268
6	289	289	289	289	289
7	834	834	834	828	805
8	430	430	430	430	430
9	924	924	924	924	924
10	1688	1688	1688	1688	1686
11	1178	1178	1178	1178	1178
12	1801	1865	1865	1865	1829
13	1452	1452	1452	1452	1452
14	1270	1270	1270	1270	1270
15	2721	2726	2726	2667	2505
16	1720	1860	1860	1820	1820
17	27486	27589	27718	27539	27517
18	21976	21976	22502	22502	21508
19	23743	23743	24019	24019	23943

20	31269	32893	32893	32893	30067
21	26332	27923	27923	27923	27790

Fonte: <sup>1</sup>Encontrado por Beasley (1985). <sup>2</sup>Encontrado por Alvarez-Valdes, Parreño e Tamarit (2005).

Ao se analisar os dados contidos na Tabela 12, nota-se que, na segunda coluna, são apresentados os dados obtidos por Beasley (1985), que utilizou como técnica de solução um método exato; na terceira coluna, os resultados obtidos por Alvarez-Valdes, Parreño e Tamarit (2005), que utilizaram a meta-heurística GRASP para a resolução do problema em estudo; na quarta coluna, os resultados obtidos pelo Sistema I, que utilizou o modelo matemático proposto por Hadjiconstantinou e Christofides (1995). Na sequência, são apresentados os resultados obtidos pelo Sistema II e Sistema III, com os algoritmos especializados propostos pela autora na resolução do problema em estudo.

Comparando-se os resultados, observa-se que o Sistema I apresentou melhores soluções em relação a Beasley (1985), onde se compara dois métodos exatos aplicados na resolução do mesmo problema com as mesmas instâncias testadas. Os resultados obtidos pelo Sistema I são considerados soluções ótimas e serviram de parâmetro para avaliar as meta-heurísticas especializadas.

Os resultados apresentados pela meta-heurística GRASP, de Alvarez-Valdes, Parreño e Tamarit (2005), são muito próximos dos resultados ótimos com o Sistema I, somente em três casos de teste obteve-se resultados inferiores (casos 17, 18 e 19).

Em relação aos resultados encontrados pelo Sistema II, proposto pela autora, utilizando-se meta-heurística especializada, também se constatou que os resultados são próximos das soluções ótimas do Sistema I, portanto são resultados de boa qualidade, porém apresentam seis casos de teste com resultados inferiores ao ótimo (casos 1, 3, 7, 15, 16, 17).

Comparando-se os resultados dos casos de teste de Alvarez-Valdes, Parreño e Tamarit (2005) com os resultados obtidos pelo Sistema II, proposto pela autora, nota-se que o algoritmo de Alvarez-Valdes, Parreño e Tamarit (2005) se mostrou melhor em instâncias menores, como são os casos: 1, 3, 7, 15 e 16. Nesses casos, as placas têm dimensões que variam de 10 x 10 até 40 x 70, e a quantidade de tipos de peças varia de 5 a 20 tipos de peças, essas instâncias são consideradas de pequeno e médio porte.

Nos casos 18 e 19, que são considerados de grande porte, com placas de dimensões 100 x 100 e quantidade de tipos de peças chegando a 30 tipos diferentes, o Sistema II, proposto pela autora, mostrou-se melhor que o algoritmo de Alvarez-Valdes, Parreño e Tamarit (2005), apresentando, portanto, sua contribuição e melhoria para a resolução do problema de corte

bidimensional não guilhotinado.

Atribui-se à estratégia de formação de blocos, adotada pelos algoritmos especializados propostos neste trabalho, o mau desempenho do Sistema II, com casos de testes menores. A formação de bloco prioriza a aglomeração do maior número de peças do mesmo tipo formando um bloco e, em alguns dos casos onde o Sistema II teve desempenho pior, verificou-se que, tanto no Sistema II como no Sistema I, os tipos de peças cortadas na placa se repetiam, no entanto com uma quantidade menor de exemplares no Sistema I (onde o desempenho foi melhor) e com mais exemplares cortados na placa no Sistema II (onde o desempenho foi pior). Mas em casos de teste maiores, onde o Sistema II obteve desempenho melhor, o motivo do bom desempenho pode ser justamente a formação de blocos.

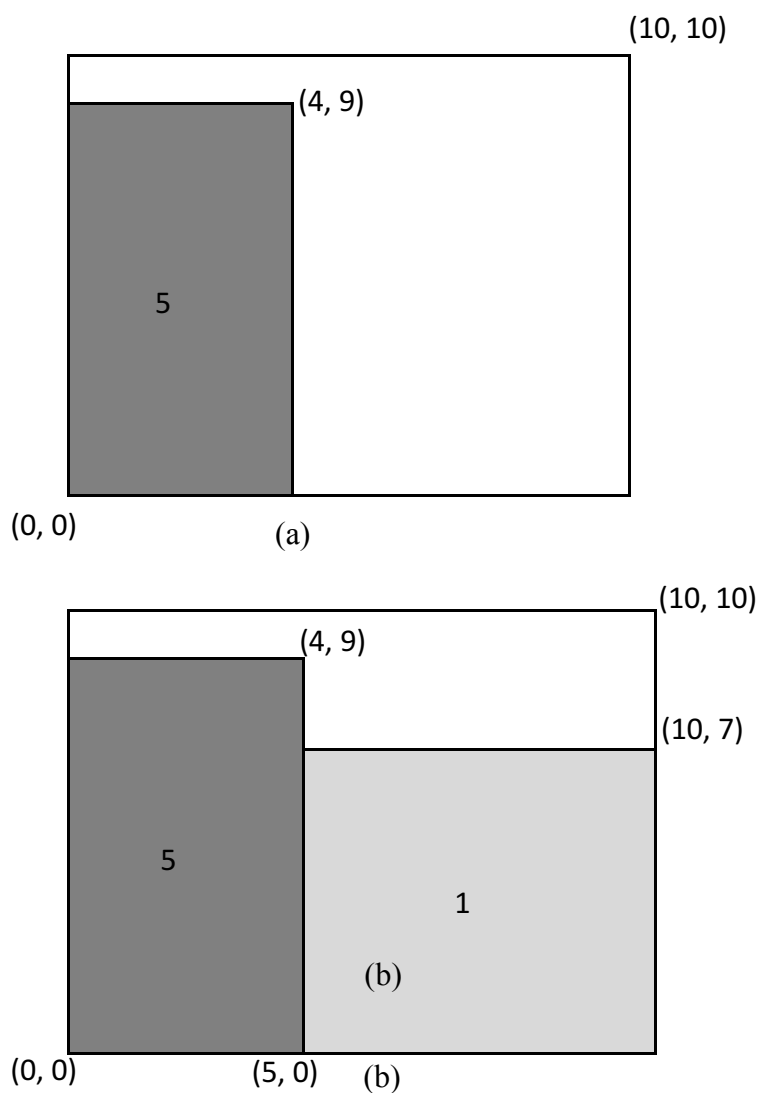
No caso 3, por exemplo, usando-se o Sistema I foi obtido o seguinte resultado: valor total das peças cortadas: 164, com as seguintes peças cortadas na placa: 1, 2, 4, 4, 5. Analisando-se o mesmo caso 3, porém usando o Sistema II, verifica-se que o valor encontrado foi menor: 156, com as seguintes peças: 5 (com dois exemplares) e 1 (com dois exemplares), blocos cortados nas seguintes coordenadas: bloco 5, com coordenadas ((1, 1),(4, 9)), e bloco 1, com coordenadas ((5, 1),(10, 7))

Simulando-se o que acontece com a placa ao serem cortadas as peças resultantes do caso 3, no Sistema II, verifica-se que, ao cortar a peça 5, com dois exemplares, e a peça 1, também com dois exemplares, torna-se impossível o corte de peças do tipo 2 e 4 como ocorre no Sistema I.

A Figura 36 mostra como fica a placa após o corte dos blocos nas coordenadas dadas pelo caso 3, do Sistema II. A Figura 36(a) exhibe a placa após o corte do bloco da peça tipo 5, com dois exemplares, e a Figura 36 (b) mostra como fica a placa após o corte do bloco da peça tipo 1, com dois exemplares.

Analisando-se a Figura 36, sob o ponto de vista da representação para solução do problema, proposta pela autora na seção 3.5, nota-se que este tipo de representação torna o trabalho de corte das peças na placa mais simples, pois adota o sistema de coordenadas de blocos. Portanto, os algoritmos especializados, propostos pela autora com a proposta de representação da solução do problema, podem ser adotados como planejadores de corte por uma eventual indústria que trabalhe com corte de peças em uma placa de matéria-prima.

**Figura 36** - Placa Após o Corte dos Blocos na Sequência de Peças Dada pelo Caso 3 do Sistema II. (a) Após o Corte do Bloco da Peça Tipo 5 com Dois Exemplos. (b) Após o Corte do Bloco da Peça Tipo 1 com Dois Exemplos

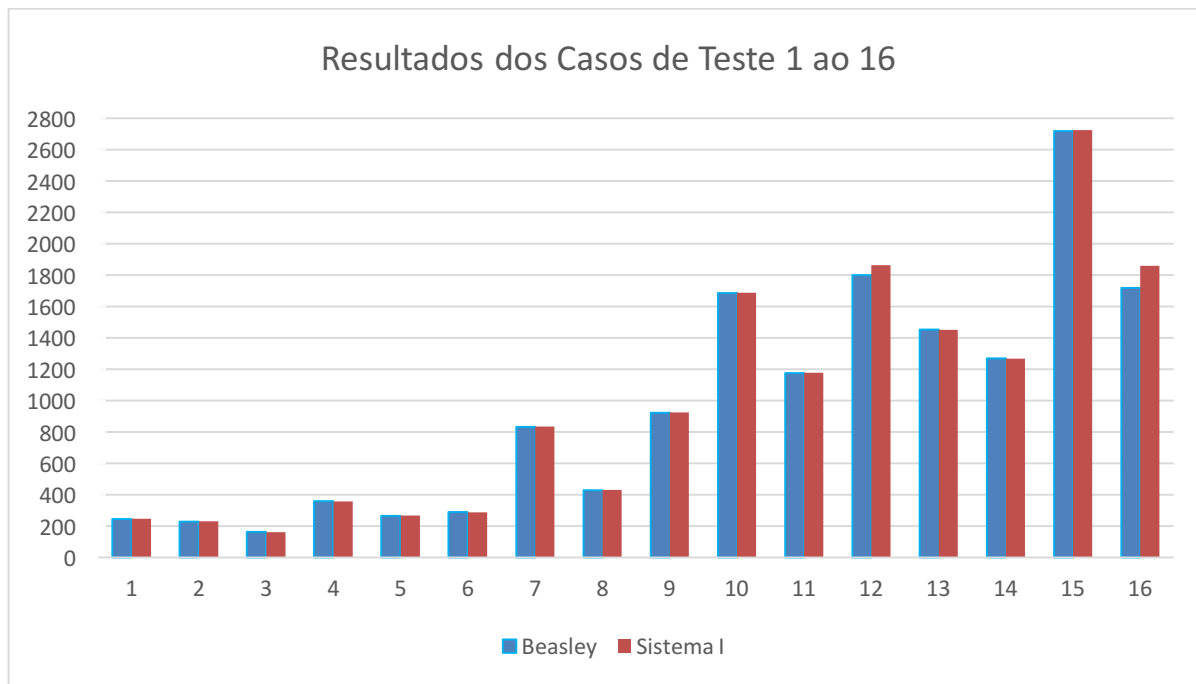


Fonte: Elaborado pela autora.

Com relação ao algoritmo especializado usado pelo Sistema III, os resultados encontrados não foram bons comparados aos outros algoritmos meta-heurísticos analisados neste trabalho. Somente em oito casos (4, 5, 6, 8, 9, 11, 13, 14) seus resultados se mantiveram iguais aos outros algoritmos, pois nos demais casos o Sistema III se mostrou inferior.

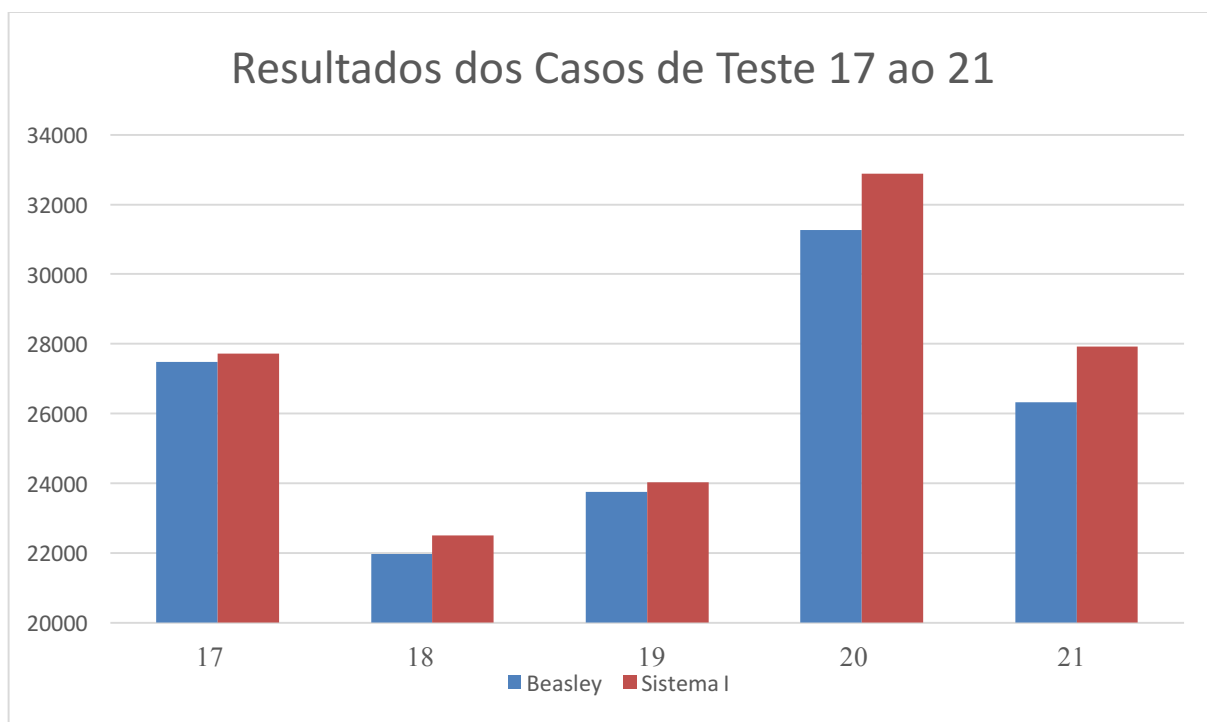
As Figuras 37 e 38, a seguir, demonstram a performance entre o método exato de Beasley (1985) e o Sistema I. A Figura 37 exibe a comparação dos resultados dos casos de teste 1 ao 16 e a Figura 38 exibe a comparação dos resultados dos casos de teste 17 ao 21.

**Figura 37** - Comparação de Performance dos Métodos Exatos Beasley (1985) e Sistema I – Casos 1 ao 16



Fonte: Elaborado pela autora.

**Figura 38** - Comparação de Performance dos Métodos Exatos Beasley (1985) e Sistema I – Casos 17 ao 21



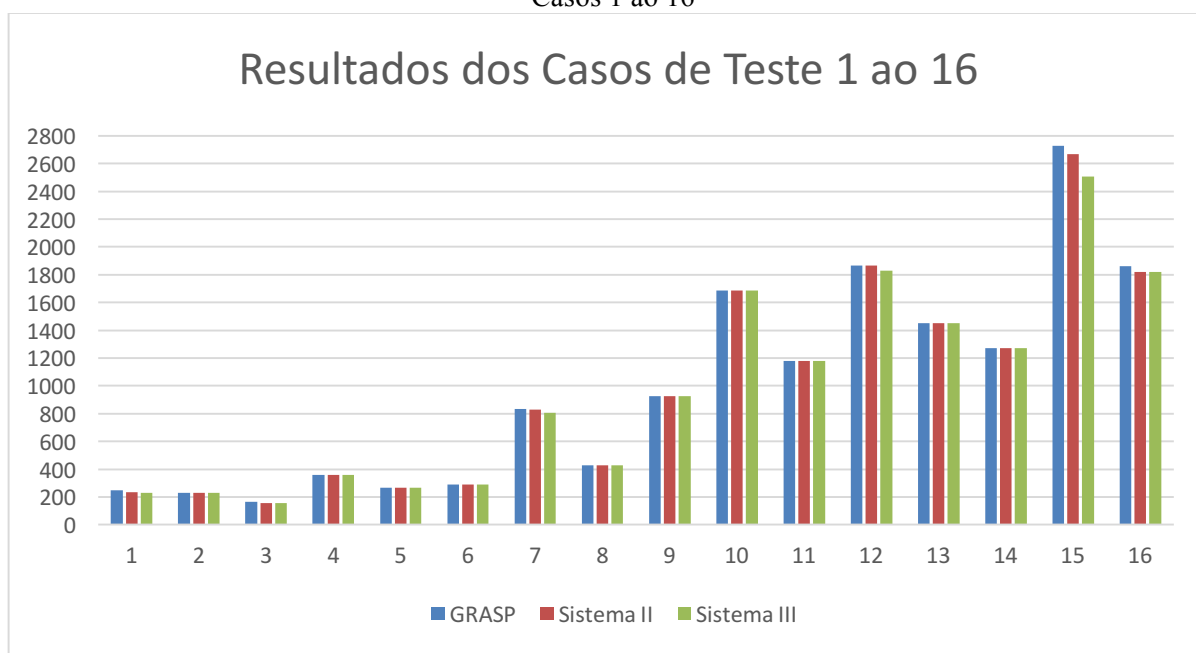
Fonte: Elaborado pela autora.

Nas Figuras 39 e 40, é possível comparar a performance entre as meta-heurísticas



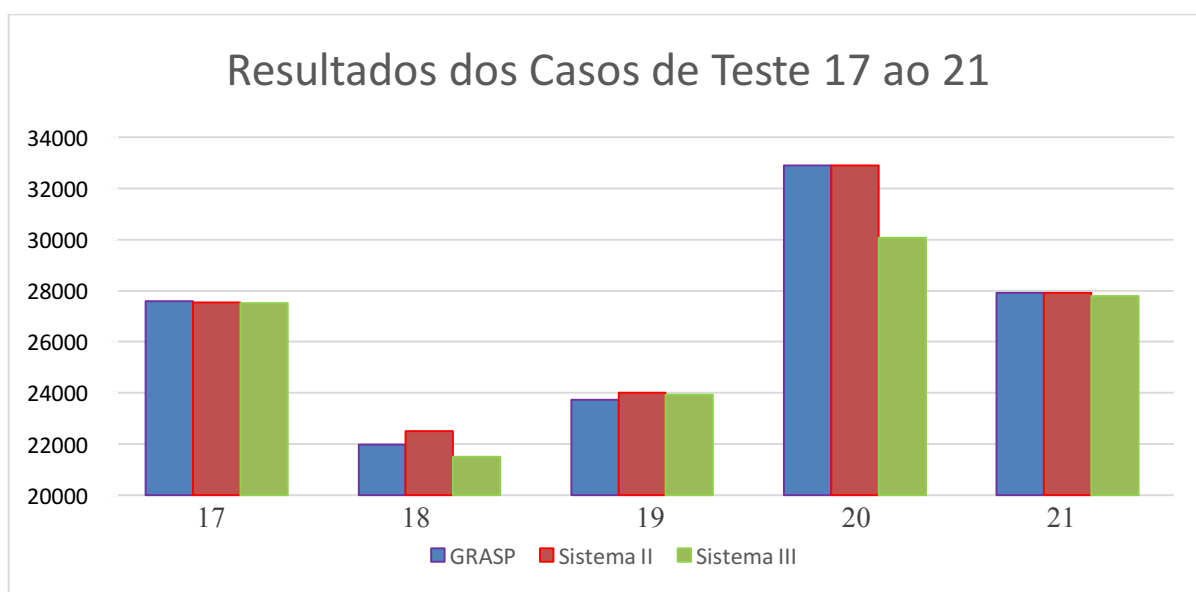
propostas por Alvarez-Valdes, Parreño e Tamarit (2005) e as propostas pela autora no Sistema II e Sistema III. A Figura 39 exibe a comparação dos resultados dos casos de teste 1 ao 16, e a Figura 40 exibe a comparação dos resultados dos casos de teste 17 ao 21.

**Figura 39** - Comparação de Performance Entre os Algoritmos GRASP, Sistema II e Sistema III – Casos 1 ao 16



Fonte: Elaborado pela autora.

**Figura 40** - Comparação de Performance Entre os Algoritmos GRASP, Sistema II e Sistema III – Casos 17 ao 21



Fonte: Elaborado pela autora.

Como já mencionado anteriormente, a autora propôs um algoritmo genético de chaves

aleatórias viciadas especializado no problema de corte bidimensional não guilhotinado com orientação livre das peças (Sistema IV), portanto um outro problema, com um espaço de busca maior do que o problema com orientação fixa. Por este motivo, os resultados, considerando-se orientação fixa das peças, não podem ser comparados diretamente com os resultados, considerando a orientação livre das peças. No entanto, a comparação é válida se for observado que os resultados do Sistema IV devem ser maiores ou iguais aos resultados do Sistema II.

Não foram encontradas, na literatura especializada, publicações utilizando o problema de corte bidimensional não guilhotinado, considerando a orientação livre das peças a serem cortadas em uma placa, portanto a comparação realizada na Tabela 13 mostra apenas que, em alguns casos, o mesmo valor foi encontrado tanto para o problema que se considerou orientação fixa quanto nos problemas que se considerou orientação livre das peças.

**Tabela 13** - Comparação de Resultados Computacionais Considerando Orientação Fixa e Livre

Comparação dos Valores Totais das Peças Cortadas com Orientação Fixa e Livre						
Caso de Teste	Beasley <sup>1</sup>	GRASP <sup>2</sup>	Sistema I	Sistema II	Sistema III	Sistema IV
1	247	247	247	234	230	259
2	230	230	230	230	228	250
3	164	164	164	156	156	193
4	358	358	358	358	358	370
5	268	268	268	268	268	268
6	289	289	289	289	289	298
7	834	834	834	828	805	856
8	430	430	430	430	430	430
9	924	924	924	924	924	924
10	1688	1688	1688	1688	1686	1786
11	1178	1178	1178	1178	1178	1272
12	1801	1865	1865	1865	1829	1932
13	1452	1452	1452	1452	1452	1452
14	1270	1270	1270	1270	1270	1431
15	2721	2726	2726	2667	2505	2793
16	1720	1860	1860	1820	1820	1840
17	27486	27589	27718	27539	27517	28090
18	21976	21976	22502	22502	21508	22852
19	23743	23743	24019	24019	23943	24811

20	31269	32893	32893	32893	30067	32893
21	26332	27923	27923	27923	27790	27983

Fonte: <sup>1</sup>Encontrado por Beasley (1985). <sup>2</sup>Encontrado por Alvarez-Valdes, Parreño e Tamarit (2005).

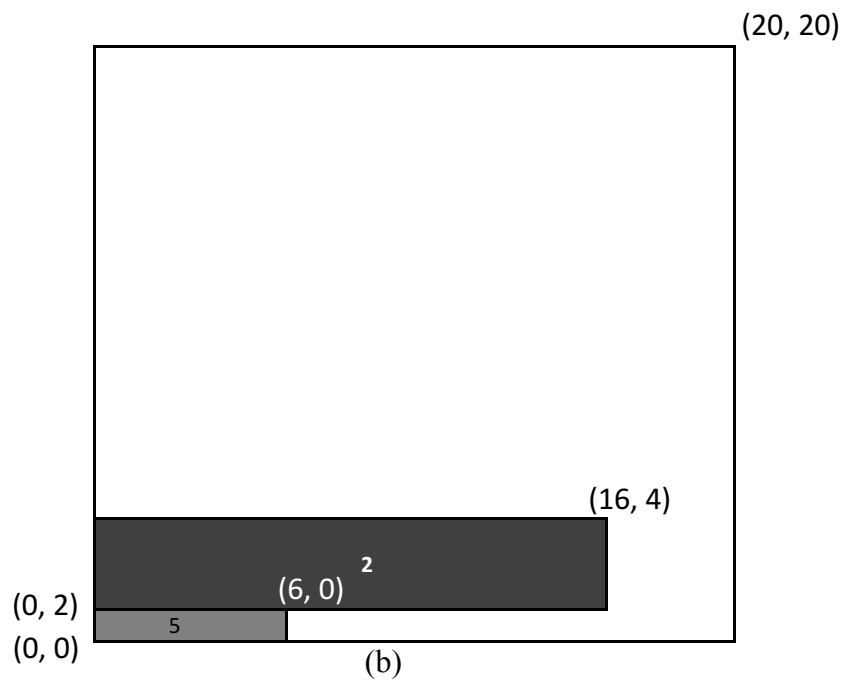
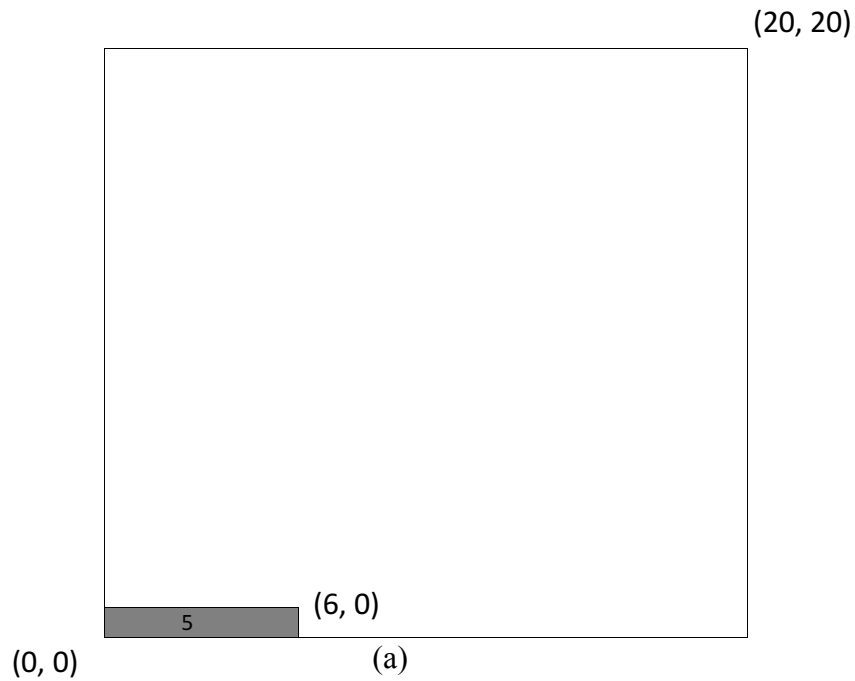
Analisando-se os resultados contidos na Tabela 13, é possível notar que existem casos de testes onde obteve-se o mesmo valor do total de peças cortadas na placa, mas quando observadas as outras informações que compõem o padrão de corte, verifica-se que são compostas pelo mesmo conjunto de tipos de peças, mudando apenas a ordem entre elas e, conseqüentemente, as coordenadas dos blocos, logo percebe-se que há mais de uma solução para o mesmo caso de teste, como acontece no caso de teste 9, por exemplo.

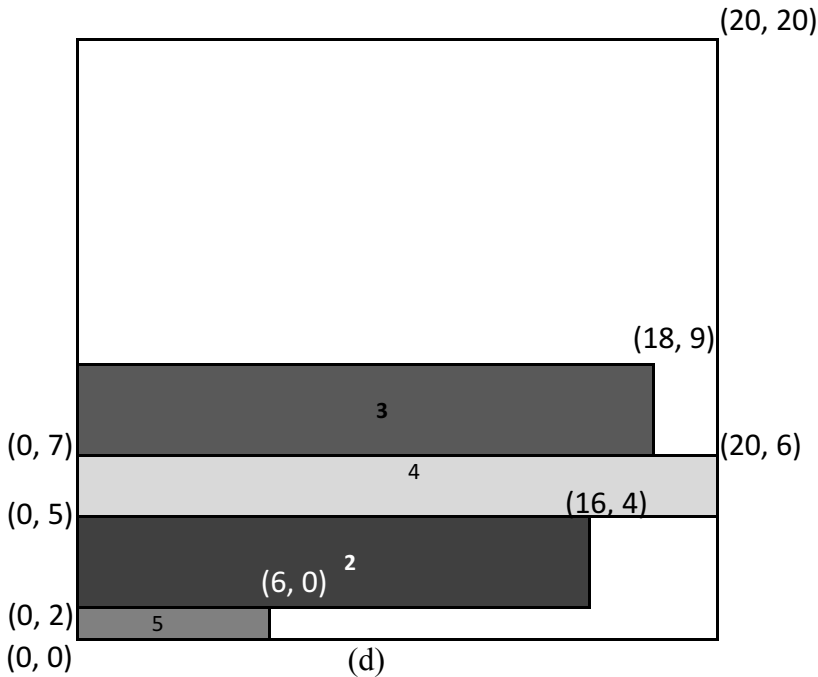
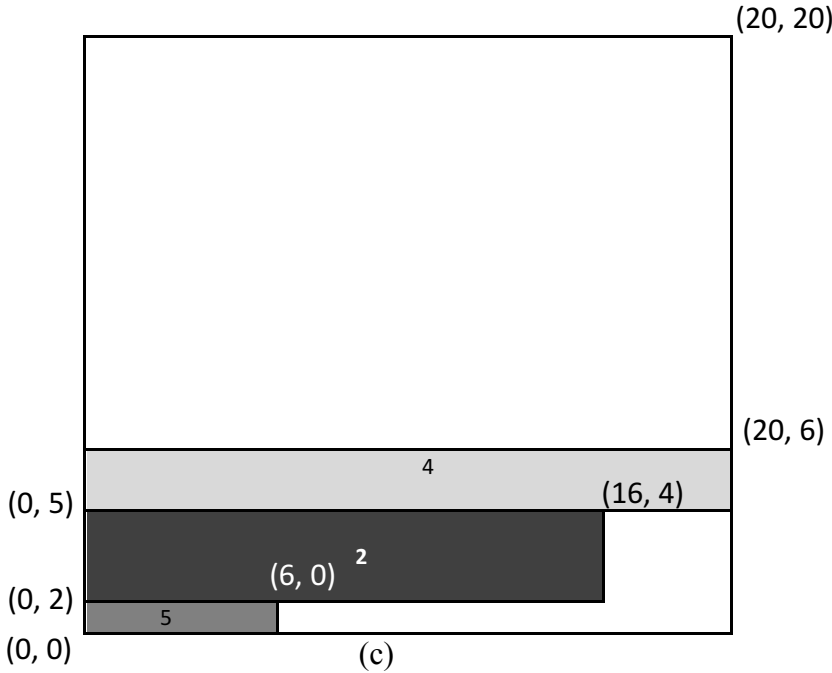
Ainda sobre o mesmo assunto, é possível notar que nos casos de teste 5, 8 e 13 foram encontrados os mesmos resultados, independentemente da técnica adotada para resolver o problema de corte bidimensional não guilhotinado. Nesses casos, existe a possibilidade de o resultado ser ótimo global, seja a peça de orientação fixa ou livre.

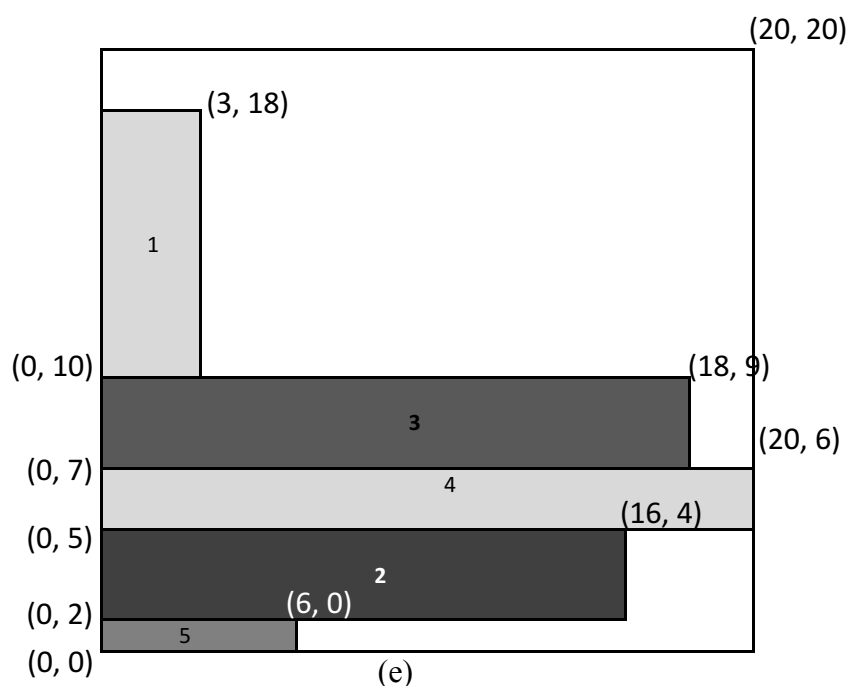
O caso de teste 8 é particular, pois todas as peças disponíveis para serem cortadas são utilizadas no corte da placa, isto é, o total das áreas de todas as peças disponíveis para corte é menor do que a área da placa, portanto todas as peças foram utilizadas. Assim, para qualquer método exato ou algoritmo especializado, obtém-se a mesma solução. A Figura 41 ilustra o caso de teste 8, que obteve como solução aplicando o Sistema II o valor total das peças cortadas: 430; a ordem de corte das peças: 5, 2, 4, 3, 1; a quantidade de peças que formam cada bloco da lista de ordem: 2, 1, 1, 1, 3; e coordenadas de bloco: bloco5 ((1, 1),(6, 1)), bloco2 ((1, 2),(16, 4)), bloco4 ((1, 5),(20,6)), bloco3 ((1, 7),(18, 9)) e bloco1 ((1, 10),(3, 18)), conforme a representação de solução proposta pela autora.

Com a seqüência de cortes apresentadas pelas Figuras 41(a), (b), (c), (d) e (e), demonstrou-se também como a representação de solução proposta pela autora auxilia no planejamento de corte e, efetivamente, no corte das peças na placa. Como já discutido na seção 3.5, se a proposta de solução é representada conforme a autora sugere, o padrão de corte pode ser interpretado por pessoas inexperientes dentro da indústria de corte, pois, certamente, não haverá problemas.

**Figura 41** - Planejamento de Corte do Caso de Teste 8. (a) Corte do Bloco da Peça Tipo 5 com Dois Exemplares. (b) Corte do Bloco da Peça Tipo 2 com Um Exemplar. (c) Corte do Bloco da Peça Tipo 4 com Um Exemplar. (d) Corte do Bloco da Peça Tipo 3 com Um Exemplar. (e) Corte do Bloco da Peça Tipo 1 com Três Exemplares







Fonte: Elaborado pela autora.

O algoritmo simula como será o corte das peças e somente o realiza, na placa, depois que toda a sequência de corte estiver pronta. Se não for assim, pode-se cortar uma peça em um retângulo livre da placa e, após isso, não ter a possibilidade de realocar a peça em outro local para algum ganho na otimização do processo, pois o corte já se consumiu. Essa é a importância da representação da proposta de solução da autora, ou seja, somente depois da sequência de corte finalizada, começa-se a cortar as peças na placa. E a representação da proposta de solução é importante neste momento, onde uma pessoa ou uma máquina terá que interpretar a sequência de peças a serem cortadas. Pela proposta da autora, a representação da solução traz todas as informações necessárias para finalizar o processo, não precisando conhecer as regras e especificidades do algoritmo para saber onde cortar a peça na placa. Portanto, a representação proposta pela autora, além de poupar espaço de armazenamento, por não estocar todas as informações sobre retângulos livres a cada corte de peça, também agiliza o processo de corte, porque não necessita da interpretação do algoritmo decodificador acompanhando o processo de finalização de cortes na placa. Os cortes usando a representação proposta pela autora acontecem de maneira natural e independentemente do algoritmo decodificador.

## 7 CONCLUSÕES

Apresentou-se, neste trabalho, uma ampla abordagem sobre o problema de corte bidimensional não guilhotinado, um problema clássico e bastante importante na área da otimização, com aplicações reais na indústria de corte de matéria-prima. Foram apontadas as principais dificuldades na resolução desse problema, as quais estão relacionadas com a natureza combinatória do processo de corte de peças em uma placa que normalmente leva a um número explosivo de alternativas, além de apresentar-se como um problema complexo em relação à sua configuração ou variantes, já que existem muitas maneiras de configurá-lo: Com quantos tipos de peças irá trabalhar; utilizar limitante superior e inferior; tamanho das peças; utilizar peças heterogêneas ou homogêneas; as peças terão orientação livre ou fixa.

Realizou-se uma extensa pesquisa na literatura especializada, buscando encontrar quais técnicas de solução foram aplicadas na resolução do problema de corte bidimensional não guilhotinado, as meta-heurísticas foram as mais utilizadas.

Durante a busca por publicações relevantes na área, notou-se que houve uma intensificação nas publicações sobre o problema de corte bidimensional não guilhotinado e suas variantes na década de 90, mas, ao final dos anos 2000, percebeu-se que as publicações sobre o assunto foram cada vez menores. Três artigos publicados na área e tidos como referência foram selecionados e utilizados como base para os estudos e as propostas do presente trabalho.

Para a resolução do problema em estudo, este trabalho fez uso de duas meta-heurísticas especializadas: algoritmo genético de chaves aleatórias viciadas e algoritmo RVNS.

Nos dois algoritmos especializados, utilizaram-se estratégias específicas para a resolução do problema, sendo a primeira estratégia a formação de blocos de peças do mesmo tipo; a segunda estratégia, o escaneamento de retângulos livres na placa, procedimento proposto pela autora como um novo tipo de mapeamento de espaços livres e, por último, as estruturas de vizinhança utilizadas no RVNS especializado, que tinham como objetivo a decomposição de blocos de peças do mesmo tipo e trocas nas ordens das peças cortadas na placa.

Conclui-se que, por apresentarem combinações de estratégias ainda não testadas para a resolução do problema em estudo, os algoritmos propostos neste trabalho são considerados inéditos e contribuem com a literatura especializada.

Um modelo matemático foi programado em AMPL usando o solver CPLEX para auxiliar nas comparações entre os resultados oriundos do método exato e os resultados obtidos pelos algoritmos especializados propostos pelo presente trabalho, além de compará-los aos resultados de trabalhos já publicados e utilizados como referência na literatura, tornando mais

simples conferir a qualidade de cada um dos resultados. Com os resultados dos testes oriundos do modelo exato, conclui-se que por meio deste, os resultados são exatos sendo superiores no quesito otimalidade, porém demanda um maior tempo de execução, perdendo em comparação com os outros sistemas testados no quesito desempenho. Portanto deve-se considerar ao escolher entre qual sistema adotar, qual quesito tem maior prioridade na prática industrial. Outro ponto importante que deve ser lembrado é que utilizando o modelo matemático para a resolução do problema não se pode trabalhar com peças orientadas livremente, existe esta restrição que deve ser observada e pode ser pontuada como negativa do ponto de vista da indústria.

Utilizou-se vinte e um casos de teste retirados da literatura para conferir o desempenho dos algoritmos especializados propostos neste trabalho, dando maior credibilidade aos testes.

Conclui-se que o algoritmo genético de chaves aleatórias viciadas especializado proporcionou melhores resultados em relação a aplicação do algoritmo RVNS especializado para a resolução do problema de corte bidimensional não guilhotinado.

Quando comparados os resultados obtidos pelo algoritmo genético especializado proposto neste trabalho com os resultados disponíveis na literatura, conclui-se que o algoritmo proposto pela autora superou em dois casos de teste os resultados já publicados, conferindo outro ineditismo ao trabalho apresentado.

Testou-se um quarto algoritmo especializado, configurado para que as peças girassem se fosse necessário. Os resultados dos vinte e um testes neste caso foram iguais ou superiores aos já publicados na literatura, porém considerando as peças com orientação fixa. Algoritmos considerando peças com orientação livre não foram encontrados durante a pesquisa, portanto pode-se considerar como uma outra contribuição deste trabalho para a literatura especializada.

Finalizando as contribuições deste trabalho para a resolução do problema de corte bidimensional não guilhotinado, analisou-se diferentes propostas de representação de solução para o problema, e se propôs uma nova representação de proposta de solução, baseada em coordenadas de blocos. A proposta da autora para representar uma solução do problema revelou-se como uma nova contribuição para a área e confere outro ineditismo do trabalho.

Como sugestão para trabalhos futuros, a Autora destaca:

- Melhorar a definição de vizinhança no algoritmo RVNS especializado;
- Considerar a possibilidade de estender para três dimensões a resolução do problema, empregando os algoritmos especialistas desenvolvidos durante este trabalho;
- Implementar os algoritmos especializados considerando computação paralela.



## REFERÊNCIAS

- ALVAREZ-VALDES, R.; PARREÑO, F.; TAMARIT, J. M. A GRASP algorithm for constrained two-dimensional non-guillotine cutting problems. **Journal of the Operational Research Society**, Oxford, n. 56, p. 414–425, 2005.
- ALVAREZ-VALDES, R.; PARREÑO, F.; TAMARIT, J. M. A tabu search algorithm for a two-dimensional non-guillotine cutting problems. **European Journal of Operational Research**, Amsterdam, n.183, p. 1167–1182, 2007.
- ALVAREZ-VALDES, R.; PARREÑO, F.; TAMARIT, J. M. A GRASP/Path Relinking algorithm for two-and three-dimensional multiple bin-size bin packing problems. **Computers & Operations Research**, New York, n. 40, p. 3081–3090, 2013.
- ANDRADE, R.; BIRGIN, E. G.; MORABITO, R. Two-stage two-dimensional guillotine cutting stock problems with usable leftover. **International Transactions in Operational Research**, Oxford, v. 23, n. 1-2, p. 121-145, jan./mar. 2016.
- AYADIA, O.; BARKALLAH, M. An adapted particle swarm optimization approach for a 2D guillotine cutting stock problem. **Mechanics & Industry**, Les Ulis, v. 17, n. 508, p. 1-11, 2016.
- BEAN, J.C. Genetic algorithms and random keys for sequencing and optimization. **ORSA Journal on Computing**, Baltimore, v. 6, p. 154-160, 1994.
- BEASLEY, J. E. An exact two-dimensional non-guillotine cutting tree search procedure. **Operations Research**, Baltimore, n. 33, p. 49-64, 1985.
- CINTRA, G.F. et al. Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation. **European Journal of Operational Research**, Amsterdam, v. 191, n. 1, p. 61-85, nov. 2008.
- CHRISTOFIDES, N.; WHITLOCK, C. An algorithm for two-dimensional cutting problems. **Operations Research**, Baltimore, n. 25, vol. 1, p. 30–44, 1977.
- CUI, Y-P.; CUI, Y.; TANG, T. Sequential heuristic for the two-dimensional bin-packing problem. **European Journal of Operational Research**, Amsterdam, v. 240, n. 1, p. 43-53, jan. 2015.
- DARWIN, C. **Origin of Species**. London, UK: John Murray, 1859.
- DAGLI, C.H.; HAJAKBARI, A. Simulated annealing approach for solving stock cutting problem. In: International Conference on Systems, Man and Cybernetics Conference, 1990, Los Angeles. **Proceedings...** Los Angeles: IEEE, 1990. p. 221 - 223.
- DEL VALLE, A. M. et al. Heuristics for two-dimensional knapsack and cutting stock problems with items of irregular shape. **Expert Systems with Applications**, New York, v. 39, n. 16, p. 12589-12598, 2012.

DUSBERGER, F.; RAIDL, G. R. Solving the 3-staged 2-dimensional cutting stock problem by dynamic programming and variable neighborhood search. **Electronic Notes in Discrete Mathematics**, [S. l.], v. 47, p. 133-140, feb. 2015.

DYCKHOFF, H. A typology of cutting and packing problems. **European Journal of Operational Research**, Amsterdam, n. 44, p. 145-159, 1990.

FARLEY, A. A. The cutting stock problem in the canvas industry. **European Journal of Operational Research**, Amsterdam, n. 44, p. 247-255, 1990.

FAYARD, D.; ZISSIMOPOULOS, V. An approximation algorithm for solving unconstrained two-dimensional knapsack problems. **European Journal of Operational Research**, Amsterdam, n. 84, p. 618-632, 1995.

FEKETE, S. P.; SCHEPERS, J. **On more-dimensional packing iii: exact algorithms**. Berlin: Technical Report 97-290: Technical University of Berlin, 1997.

GILMORE, P.; GOMORY, R. A linear programming approach to the cutting stock problem. **Operational Research**, Baltimore, v. 9, n. 6, p. 849-859, 1961.

GLOVER, F.; KOCHENBERGER, G. A. **Handbook of metaheuristics**. Boston: Kluwer Academic Publishers, 2003.

GONÇALVES, J. F.; RESENDE, M. G. C. A Biased Random-Key Genetic Algorithm for a 2D and 3D Bin Packing Problem. **International Journal of Production Economics**, Amsterdam, n. 145, p. 500-510, 2013.

GOULIMIS, C. Optimal solutions for the cutting stock problem. **European Journal of Operational Research**, Amsterdam, n. 44, p. 197-208, 1990.

HADJICONSTANTINOU, E.; CHRISTOFIDES, N. An exact algorithm for general, orthogonal, two-dimensional knapsack problems. **European Journal of Operational Research**, Amsterdam, n. 83, p. 39-56, 1995.

HAESSLER, R. W.; SWEENEY, P. E. Cutting stock problems and solution procedures. **European Journal of Operational Research**, Amsterdam, n. 54, p. 141-150, 1991.

HANSEN, P.; MLADENOVIĆ, N. Variable neighborhood search: Principles and Applications. **European Journal of Operational Research**, Amsterdam, n. 130, p. 449-467, 2001.

HANSEN, P.; MLADENOVIĆ, N. A tutorial on variable neighborhood search. **Les Cahiers du GERAD**, Montreal, v. 46, n. G, p. 1-23, 2003a.

HANSEN, P.; MLADENOVIĆ, N. Variable neighborhood search. In: GLOVER, F.; KOCHENBERGER, G. A. (Eds.). **Handbook of metaheuristics**. Dordrecht: Kluwer, 2003b. p. 145-184.

HEALY, P.; CREA VIN, M.; KUUSIK, A. An optimal algorithm for placement rectangle. **Operational Research**, Baltimore, n. 24, p. 73-80, 1999.

HOLLAND, J. H. **Adaptation in natural and artificial systems**. Ann Arbor, MI: Univ. Michigan Press, 1975.

JAKOBS, S. On genetic algorithms for the packing of polygons. **European Journal of Operational Research**, Amsterdam, n. 88, p. 165-181, 1996.

LAI, K. K.; CHAN, J. W. M. Developing a Simulated Annealing Algorithm for the Cutting Stock Problem. **Computers & Industrial Engineering**, New York n. 32, p. 115-127, 1997.

LIU, D.; TENG, H. An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles. **European J. of Operational Research**, Amsterdam, n. 112, p. 413-420, 1999.

MALAGUTI, E.; DURÁN, R. M.; TOTH, P. Approaches to real world two-dimensional cutting problems. **Omega**, Elmsford v. 47, p. 99-115, 2014.

MLADENović, N. A variable neighborhood algorithm-a new metaheuristic for combinatorial optimization. In: **Papers presented at Optimization Days**. [S. l.: s. n.], 1995.

MLADENović, N. A variable neighborhood algorithm - a new metaheuristic for combinatorial optimization. In: **OPTIMIZATION DAYS, 12.**, 2012, Montreal. **Abstracts...** Montreal: HEC Montréal, 2012.

MLADENović, N.; HANSEN, P. Variable neighborhood search. **Computers Ops. Research**, New York, v. 11, n. 24, p. 1097-1100, 1997.

MORABITO, R. N.; ARENALES, M. N.; ARCARO, V. F. An and-or-graph approach for two-dimensional cutting problems. **European Journal of Operational Research**, Amsterdam, n. 58, p. 263-271, 1992.

MORABITO, R. N.; ARENALES, M. N. An And/Or-graph approach to the solution of two-dimensional non-guillotine cutting problems. **European Journal of Operational Research**, Amsterdam, n. 84, p. 599-617, 1995.

MORABITO, R. N.; PUREZA, V. Geração de padrões de cortes bidimensionais guilhotinados restritos via programação dinâmica e busca em grafo-e/ou. **Revista Production**, São Paulo, v. 17, n. 1, p. 33-51, 2007.

OR-LIBRARY. **Operations Research (OR) problems**. [S. l.], 2017. Online. Disponível em: <<http://people.brunel.ac.uk/~mastjjb/jeb/info.html>>. Acesso em: 20 abr. 2017.

PISINGER, D.; SIGURD, M. Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem. **INFORMS Journal on Computing**, Linthicum, v. 19, n. 1, p. 36-51, 2007.

POSSAGNOLO, L. H. F. M. **Reconfiguração de sistemas de distribuição operando em vários níveis de demanda através de uma meta-heurística de Busca em Vizinhança Variável**. 2015. 178 f. Dissertação (Mestrado) - Universidade Estadual Paulista, Faculdade de Engenharia de Ilha Solteira, Ilha Solteira, 2015.

RIBEIRO, C. C. **Metaheuristics and applications**. Estoril: Advanced School on Artificial Intelligence, 1996.

RODRIGUES, L. L. **Um algoritmo genético para o problema de carregamento de container**. 2005. 105 f. Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2005.

ROMERO, R.; MANTOVANI, J. R. S. Introdução à meta-heurística. In: Congresso Temático de Dinâmica e Controle da SBMAC, 3., 2004, Ilha Solteira-SP. **Anais...** Ilha Solteira, 2004. p. 1-33.

RUSSO, M.; SFORZA, A.; STERLE, C. An exact dynamic programming algorithm for large-scale unconstrained two-dimensional guillotine cutting problems. **Computers & Operations Research**, New York, v. 50, p. 97-114, 2014.

SILVA, E.; ALVELOS, F.; VALÉRIO DE CARVALHO, J. M. An integer programming model for two- and three-stage two-dimensional cutting stock problems. **European Journal of Operational Research**, Amsterdam, n. 205, p. 699-708, 2010.

SOARES, G. L. **Algoritmos genéticos: estudos, novas técnicas e aplicações**. [S. l.], 1997. Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal de Minas Gerais, Belo Horizonte, 1997.

SPEARS, W. M.; DEJONG, K. A. On the virtues of parameterized uniform crossover. In: INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, 4., 1991, San Diego. **Proceedings**. San Diego: University of California, 1991. p. 230-236.

TEMPONI, E. C. C. **Uma proposta de resolução do problema de corte bidimensional via abordagem metaheurística**. 2007. 100 f. Dissertação (Mestrado) - Centro Federal de Educação Tecnológica de Minas Gerais, Belo Horizonte, 2007.

TSAI, R. D.; MALSTROM, E. M.; MEEKS, H. D. A two-dimensional palletizing procedure for warehouse loading operations. **IIE Trans.**, [S. l.], n. 20, p. 418-425, 1988.

VENDRAMINI, E. **Otimização do problema de carregamento de container usando uma metaheurística eficiente**. 2007. 115 f. Dissertação (Mestrado em Engenharia Elétrica) – Universidade Estadual Paulista, Ilha Solteira, 2007.

WANG, P. Y. Two algorithms for constrained two-dimensional cutting stock problems. **Operations Research**, v. 3, n. 31, p. 573-586, 1983.

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

### APÊNDICE A – Dados da pesquisa obtidos na OR-Library (2017, online).

Para um melhor entendimento sobre os dados da Tabela A1 até Tabela A 21, são descritos, a seguir, os cabeçalhos das colunas.

$R = (\text{comprimento} \times \text{altura})$ : são as dimensões de comprimento e altura da placa para cada caso de teste.

Peça: representa os tipos de peças disponíveis em cada caso de teste, onde a quantidade de tipos de peça, para cada caso, de teste vai de 1 a  $m$ ;

$C_i$ : comprimento da peça  $i$ ;

$A_i$ : altura da peça  $i$ ;

$P_i$ : limitante inferior da peça  $i$ ;

$Q_i$ : limitante superior da peça  $i$ , onde  $Q_i$  também representa a quantidade de peças  $i$  disponíveis para o corte;

$V_i$ : valor da peça  $i$ ;

**Tabela A1** - Dados das peças e placa do caso de teste 1

Placa Retangular: $R = (10 \times 10)$					
Peça	$C_i$	$A_i$	$P_i$	$Q_i$	$V_i$
1	3	2	0	2	7
2	7	2	0	3	20
3	4	5	0	2	60
4	4	1	0	2	9
5	1	10	0	1	14
6	8	4	0	1	79
7	4	2	0	2	11
8	3	7	0	3	52
9	6	2	0	3	13
10	9	1	0	2	21

Fonte: Dados obtidos em OR-Library (2017, online) pela autora.

**Tabela A2** - Dados das peças e placa do caso de teste 2

Placa Retangular: $R = (10 \times 10)$					
Peça	$C_i$	$A_i$	$P_i$	$Q_i$	$V_i$
1	1	10	0	3	28
2	5	3	0	2	40
3	9	3	0	3	63
4	6	1	0	3	13
5	3	8	0	3	31
6	4	1	0	1	10
7	6	3	0	2	44

Fonte: Dados obtidos em OR-Library (2017, online) pela autora.

**Tabela A3** - Dados das peças e placa do caso de teste 3

Placa Retangular: $R = (10 \times 10)$					
<i>Peça</i>	$C_i$	$A_i$	$P_i$	$Q_i$	$V_i$
1	3	7	0	2	35
2	8	2	0	2	40
3	10	2	0	1	27
4	5	4	0	3	23
5	2	9	0	2	43

Fonte: Dados obtidos em OR-Library (2017, online) pela autora.

**Tabela A4** - Dados das peças e placa do caso de teste 4

Placa Retangular: $R = (15 \times 10)$					
<i>Peça</i>	$C_i$	$A_i$	$P_i$	$Q_i$	$V_i$
1	10	3	0	2	34
2	9	3	0	1	48
3	12	2	0	3	72
4	11	3	0	1	91
5	12	3	0	3	37
6	11	1	0	3	15
7	2	10	0	1	36

Fonte: Dados obtidos em OR-Library (2017, online) pela autora.

**Tabela A5** - Dados das peças e placa do caso de teste 5

Placa Retangular: $R = (15 \times 10)$					
<i>Peça</i>	$C_i$	$A_i$	$P_i$	$Q_i$	$V_i$
1	15	2	0	2	34
2	7	3	0	1	27
3	9	1	0	1	14
4	8	3	0	1	71
5	12	2	0	2	61

Fonte: Dados obtidos em OR-Library (2017, online) pela autora.

**Tabela A6** - Dados das peças e placa do caso de teste 6

Placa Retangular: $R = (15 \times 10)$					
<i>Peça</i>	$C_i$	$A_i$	$P_i$	$Q_i$	$V_i$
1	2	9	0	1	50
2	10	2	0	3	29
3	2	4	0	1	11
4	2	8	0	2	30
5	3	8	0	1	46
6	4	1	0	1	7
7	6	4	0	3	32
8	10	3	0	1	74
9	11	2	0	1	48
10	4	5	0	1	32

Fonte: Dados obtidos em OR-Library (2017, online) pela autora.

**Tabela A7** - Dados das peças e placa do caso de teste 7

Placa Retangular: $R = (20 \times 20)$					
<i>Peça</i>	$C_i$	$A_i$	$P_i$	$Q_i$	$V_i$
1	15	3	0	1	110
2	6	6	0	3	93
3	13	1	0	2	38
4	8	7	0	2	144
5	18	5	0	3	185
6	12	4	0	1	78
7	8	3	0	1	25

Fonte: Dados obtidos em OR-Library (2017, online) pela autora.

**Tabela A8** - Dados das peças e placa do caso de teste 8

Placa Retangular: $R = (20 \times 20)$					
<i>Peça</i>	$C_i$	$A_i$	$P_i$	$Q_i$	$V_i$
1	1	9	0	3	26
2	16	3	0	1	113
3	18	3	0	1	127
4	20	2	0	1	104
5	3	1	0	2	4

Fonte: Dados obtidos em OR-Library (2017, online) pela autora.

**Tabela A9** - Dados das peças e placa do caso de teste 9

Placa Retangular: $R = (20 \times 20)$					
<i>Peça</i>	$C_i$	$A_i$	$P_i$	$Q_i$	$V_i$
1	14	2	0	1	45
2	1	5	0	1	10
3	20	4	0	2	139
4	12	3	0	3	92
5	11	8	0	2	145
6	11	6	0	2	123
7	7	9	0	1	166
8	17	5	0	1	252
9	7	14	0	2	161
10	1	7	0	3	12

Fonte: Dados obtidos em OR-Library (2017, online) pela autora.

**Tabela A10** - Dados das peças e placa do caso de teste 10

Placa Retangular: $R = (30 \times 30)$					
<i>Peça</i>	$C_i$	$A_i$	$P_i$	$Q_i$	$V_i$
1	3	23	0	3	118
2	29	5	0	1	244
3	21	2	0	3	81
4	17	11	0	3	371
5	14	7	0	2	254

6	8	5	0	2	78
7	21	8	0	1	293

Fonte: Dados obtidos em OR-Library (2017, online) pela autora.

**Tabela A11** - Dados das peças e placa do caso de teste 11

Placa Retangular: $R = (30 \times 30)$					
<i>Peça</i>	$C_i$	$A_i$	$P_i$	$Q_i$	$V_i$
1	12	21	0	1	450
2	17	21	0	1	570
3	6	6	0	1	49
4	9	15	0	1	166
5	5	12	0	1	69
6	16	6	0	1	92
7	5	4	0	1	17

Fonte: Dados obtidos em OR-Library (2017, online) pela autora.

**Tabela A12** - Dados das peças e placa do caso de teste 12

Placa Retangular: $R = (30 \times 30)$					
<i>Peça</i>	$C_i$	$A_i$	$P_i$	$Q_i$	$V_i$
1	16	6	0	1	132
2	24	1	0	3	29
3	6	28	0	3	351
4	8	23	0	3	275
5	5	1	0	3	10
6	6	26	0	2	259
7	2	30	0	2	110
8	9	11	0	3	240
9	4	30	0	1	265
10	16	13	0	1	507

Fonte: Dados obtidos em OR-Library (2017, online) pela autora.

**Tabela A13** - Dados das peças e placa do caso de teste 13

Placa Retangular: $R = (30 \times 30)$					
<i>Peça</i>	$C_i$	$A_i$	$P_i$	$Q_i$	$V_i$
1	1	30	0	3	75
2	25	7	0	1	279
3	27	9	0	3	409
4	30	2	0	3	65
5	26	7	0	3	235

Fonte: Dados obtidos em OR-Library (2017, online) pela autora.

**Tabela A14** - Dados das peças e placa do caso de teste 14

Placa Retangular: $R = (30 \times 30)$					
<i>Peça</i>	$C_i$	$A_i$	$P_i$	$Q_i$	$V_i$
1	22	21	0	1	828



2	12	21	0	1	450
3	11	18	0	1	335
4	17	21	0	1	570
5	8	21	0	1	268
6	9	17	0	1	236
7	5	18	0	1	128
8	6	6	0	1	49
9	9	15	0	1	166
10	1	19	0	1	22
11	5	12	0	1	69
12	22	21	0	1	530
13	5	6	0	1	33
14	16	6	0	1	92
15	5	4	0	1	17

Fonte: Dados obtidos em OR-Library (2017, online) pela autora.

**Tabela A15** - Dados das peças e placa do caso de teste 15

Placa Retangular: $R = (70 \times 40)$					
<i>Peça</i>	$C_i$	$A_i$	$P_i$	$Q_i$	$V_i$
1	17	9	0	1	153
2	11	19	0	4	209
3	12	21	0	3	252
4	14	23	0	4	322
5	24	15	0	3	360
6	25	16	0	4	400
7	27	17	0	2	459
8	18	29	0	3	522
9	21	31	0	3	651
10	32	22	0	2	704
11	23	33	0	3	759
12	34	24	0	2	816
13	35	25	0	2	875
14	36	26	0	1	936
15	37	27	0	1	999
16	38	28	0	1	1064
17	39	29	0	1	1131
18	41	30	0	1	1230
19	43	31	0	1	1333

Fonte: Dados obtidos em OR-Library (2017, online) pela autora.

**Tabela A16** - Dados das peças e placa do caso de teste 16

Placa Retangular: $R = (40 \times 70)$					
<i>Peça</i>	$C_i$	$A_i$	$P_i$	$Q_i$	$V_i$
1	31	43	0	4	500
2	30	41	0	2	480
3	29	39	0	4	460
4	28	38	0	4	440
5	27	37	0	3	420

6	26	36	0	4	410
7	25	35	0	3	400
8	24	34	0	4	380
9	33	23	0	4	360
10	22	32	0	3	340
11	31	21	0	3	320
12	29	18	0	3	300
13	17	27	0	2	280
14	15	24	0	2	240
15	16	25	0	4	260
16	15	24	0	1	240
17	23	14	0	4	220
18	21	12	0	3	180
19	19	11	0	4	160
20	9	17	0	1	140

Fonte: Dados obtidos em OR-Library (2017, online) pela autora.

**Tabela A17** - Dados das peças e placa do caso de teste 17

Placa Retangular: $R = (100 \times 100)$					
<i>Peça</i>	$C_i$	$A_i$	$P_i$	$Q_i$	$V_i$
1	4	90	0	5	838
2	22	21	0	2	521
3	22	80	0	3	4735
4	1	88	0	5	181
5	6	40	0	5	706
6	100	9	0	5	2538
7	46	14	0	3	1349
8	10	96	0	1	1685
9	70	27	0	3	5336
10	57	18	0	1	1775
11	10	84	0	1	1131
12	100	1	0	5	129
13	2	41	0	5	179
14	36	63	0	2	6668
15	51	24	0	4	3551

Fonte: Dados obtidos em OR-Library (2017, online) pela autora.

**Tabela A18** - Dados das peças e placa do caso de teste 18

Placa Retangular: $R = (100 \times 100)$					
<i>Peça</i>	$C_i$	$A_i$	$P_i$	$Q_i$	$V_i$
1	8	81	0	1	953
2	5	76	0	1	389
3	42	19	0	1	1668
4	6	80	0	1	676
5	41	48	0	1	3580
6	6	86	0	1	1416

7	58	20	0	1	3166
8	99	3	0	1	537
9	9	52	0	1	1176
10	100	14	0	1	3434
11	7	53	0	1	676
12	24	54	0	1	1408
13	23	77	0	1	2362
14	42	32	0	1	4031
15	17	30	0	1	1152
16	11	90	0	1	2255
17	26	65	0	1	3570
18	11	84	0	1	1913
19	100	11	0	1	1552
20	29	81	0	1	4559
21	10	64	0	1	713
22	25	48	0	1	1279
23	17	93	0	1	3989
24	77	31	0	1	4850
25	3	71	0	1	299
26	89	9	0	1	1577
27	1	6	0	1	12
28	12	99	0	1	2116
29	33	72	0	1	2932
30	21	26	0	1	1214

Fonte: Dados obtidos em OR-Library (2017, online) pela autora.

**Tabela A19** - Dados das peças e placa do caso de teste 19

Placa Retangular: $R = (100 \times 100)$					
<i>Peça</i>	$C_i$	$A_i$	$P_i$	$Q_i$	$V_i$
1	3	98	0	1	756
2	34	36	0	1	2712
3	100	6	0	1	1633
4	49	26	0	1	2332
5	14	56	0	1	2187
6	100	3	0	1	470
7	10	90	0	1	1569
8	23	95	0	1	4947
9	10	97	0	1	2757
10	50	47	0	1	4274
11	41	45	0	1	4347
12	13	12	0	1	396
13	19	68	0	1	3866
14	50	46	0	1	5447
15	23	70	0	1	2906
16	28	82	0	1	6032
17	12	65	0	1	1799
18	9	86	0	1	929
19	21	96	0	1	5186

20	19	64	0	1	2120
21	21	75	0	1	1629
22	45	26	0	1	2059
23	19	77	0	1	2583
24	5	84	0	1	953
25	16	21	0	1	1000
26	23	69	0	1	2900
27	5	89	0	1	1102
28	22	63	0	1	2234
29	41	6	0	1	458
30	76	30	0	1	5458

Fonte: Dados obtidos em OR-Library (2017, online) pela autora.

**Tabela A20** - Dados das peças e placa do caso de teste 20

Placa Retangular: $R = (100 \times 100)$					
<i>Peça</i>	$C_i$	$A_i$	$P_i$	$Q_i$	$V_i$
1	48	48	0	1	5145
2	6	85	0	2	874
3	100	14	0	1	2924
4	17	85	0	1	3182
5	69	20	0	1	2862
6	12	72	0	1	1224
7	5	48	0	3	531
8	1	97	0	3	249
9	66	36	0	2	6601
10	15	53	0	1	1005
11	29	80	0	3	6228
12	19	77	0	1	3362
13	97	7	0	1	907
14	7	57	0	2	473
15	63	37	0	2	6137
16	71	14	0	1	1556
17	3	76	0	3	313
18	34	54	0	1	4123
19	5	91	0	2	581
20	14	87	0	1	1999
21	62	28	0	3	5004
22	6	7	0	3	2040
23	20	71	0	1	3143
24	92	7	0	1	795
25	10	77	0	2	1460
26	99	4	0	3	841
27	14	44	0	2	1107
28	100	2	0	3	280
29	56	40	0	2	5898
30	86	14	0	1	2096
31	22	93	0	1	4411

32	13	99	0	3	3456
33	7	76	0	3	1406

Fonte: Dados obtidos em OR-Library (2017, online) pela autora.

**Tabela A21** - Dados das peças e placa do caso de teste 21

Placa Retangular: $R = (100 \times 100)$					
<i>Peça</i>	$C_i$	$A_i$	$P_i$	$Q_i$	$V_i$
1	8	81	0	3	953
2	5	76	0	4	389
3	42	19	0	4	1668
4	6	80	0	4	676
5	41	48	0	1	3580
6	6	86	0	5	1416
7	58	20	0	5	3166
8	99	3	0	5	537
9	9	52	0	5	1176
10	100	14	0	4	3434
11	7	53	0	5	676
12	24	54	0	1	1408
13	23	77	0	1	2362
14	42	32	0	5	4031
15	17	30	0	5	1152
16	11	90	0	4	2255
17	26	65	0	2	3570
18	11	84	0	3	1913
19	100	11	0	1	1552
20	29	81	0	1	4559
21	10	64	0	2	713
22	25	48	0	1	1279
23	17	93	0	4	3989
24	77	31	0	1	4850
25	3	71	0	5	299
26	89	9	0	4	1577
27	1	6	0	5	12
28	12	99	0	2	2116
29	21	26	0	5	1214

Fonte: Dados obtidos em OR-Library (2017, online) pela autora.