UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

DAVID ÁLVAREZ MARTÍNEZ

ESTUDO DOS PROBLEMAS DE CORTE E EMPACOTAMENTO

Ilha Solteira

2014

DAVID ÁLVAREZ MARTÍNEZ

ESTUDO DOS PROBLEMAS DE CORTE E EMPACOTAMENTO

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Elétrica da Faculdade de Engenharia da Universidade Estadual Paulista – Campus da Ilha Solteira, como parte dos requisitos para a obtenção do título de Doutor em Engenharia Elétrica. Área de especialização: Automatização.

Prof. Dr. RUBÉN A. ROMERO LÁZARO
Orientador

Ilha Solteira

2014

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

A473e

Alvarez Martinez, David.
Estudo dos problemas de corte e empacotamento / David Alvarez Martinez. -- Ilha Solteira: [s.n.], 2014
161 f. : il.

Tese (doutorado) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira. Área de conhecimento: Automação, 2014

Orientador: Ruben Augusto Romero Lazaro
Inclui bibliografia

1. Meta-heurísticas. 2. Problemas de corte e empacotamento. 3. Problema de carregamento do contêiner. 4. Problema da mochila.

unesp

UNIVERSIDADE ESTADUAL PAULISTA

CAMPUS DE ILHA SOLTEIRA

FACULDADE DE ENGENHARIA DE ILHA SOLTEIRA

CERTIFICADO DE APROVAÇÃO

TÍTULO: Estudo dos problemas de corte e empacotamento

AUTOR: DAVID ALVAREZ MARTINEZ

ORIENTADOR: Prof. Dr. RUBEN AUGUSTO ROMERO LAZARO

Aprovado como parte das exigências para obtenção do Título de DOUTOR 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. SILVIO ALEXANDRE DE ARAUJO
Departamento de Matemática Aplicada / Instituto de Biociências, Letras e Ciências Exatas de São José do Rio Preto

Prof. Dr. JOSÉ ELIAS CLAUDIO ARROYO
Departamento de Informática / Universidade Federal de Vicosa

Prof. Dr. RAMÓN ÁLVAREZ-VALDÉS
Universitat de València

Data da realização: 13 de maio de 2014.

AGRADECIMENTOS

Ao professor Rubén Augusto Romero Lázaro.

Aos professores Francisco Parreño Torres e Ramón Álvarez-Valdés Olaguíbel.

Aos professores do programa de pós-graduação em Engenharia Elétrica da Universidade Estadual Paulista: Marcos Julio Rider Flóres e Sergio Azevedo de Oliveira.

Aos professores do programa de pós-graduação em Estadística e Pesquisa Operacional da Universidade de Valência: Rafael Martí Cunquero e Enrique Benavent López.

Ao Banco Interamericano do Desenvolvimento e sua Aliança para o Atraso.

À Música.

RESUMO

O presente trabalho propõe uma análise sobre os problemas de corte e empacotamento com restrições práticas que representam cenários reais na indústria. Em síntese o problema de corte consiste em cortar um conjunto de peças de um determinado objeto, e o problema de empacotamento consiste em alocar um conjunto de peças dentro de um objeto. No mundo real se apresenta uma grande quantidade de variações destes problemas. Neste estudo limitamo-nos a estudar os problemas com peças e objetos com formas regulares, restringindo assim os problemas de duas dimensões ao uso de retângulos e aos problemas de três dimensões ao uso de paralelepípedos. De forma específica os problemas de corte estudados neste trabalho são o problema da mochila bidimensional (2D-SLOPP, do inglês *Two-Dimensional Single Large Object Placement Problem*) com restrições de padrão de corte; valores associados às peças; limites de exemplares por peça e orientação das peças.

O segundo problema a ser estudado, é o problema da embalagem (2D-SBSBPP, do inglês *Two-Dimensional Single Bin Size Bin Packing Problem*) com restrições de padrões de corte tipo guilhotina e restrições de orientação das peças.

Finalmente, o problema de empacotamento estudado no presente trabalho é o problema do carregamento de um único contêiner (3D-SKP ou 3D-SLOPP, do inglês *Three-Dimensional Single Knapsack Problem* e *Three-Dimensional Single Large Object Placement Problem*, respectivamente) com restrições de orientação das caixas; limites de resistência das caixas ao empilhamento; limite de peso do carregamento suportado pelo contêiner; estabilidade do padrão de carregamento e carga divida em múltiplos destinos.

Estes três problemas apresentados são de grande interesse para a indústria, graças a isto, atualmente existe uma ampla literatura especializada de trabalhos referentes a esta temática. Logo, diferentes tipos de metodologias tanto exatas como aproximadas têm sido propostas. Porém devido a complexidade dos problemas, as metodologias exatas só conseguem resolver instâncias de tamanho reservado. Sendo assim, neste estudo se apresentam representações adequadas dos problemas, utilizando diferentes métodos de solução aproximados, que estão fundamentados nas técnicas meta-heurísticas para otimização combinatória.

Dentro dos métodos de solução aproximados propostos em este trabalho são apresentados: um algoritmo híbrido de otimização de enxame de partículas, busca em vizinhança variável e algoritmos genéticos para os problemas da mochila e da embalagem e um algoritmo GRASP para o problema de carregamento do contêiner.

Palavras-chave - Meta-heurísticas. Problemas de corte e empacotamento. Problema de carregamento do contêiner. Problema da embalagem. Problema da mochila.

ABSTRACT

In this work we study the cutting and packing problems with practical constraints that represent real world scenarios of the industry. The cutting problem consists in to cut a set of pieces from an object, and the packing problem consists in to pack a set of items in an object.

In the real world there are a big number of variations of this problem. In this study we only carry out the problems where the pieces and the objects have a regular shape, bounding of that way the two-dimensional problems to use just rectangular items and the three-dimensional problems to use just parallelepiped pieces. Specifically, the cutting problems studied in this work are: the Two-Dimensional Single Knapsack Problem, taking into account cutting pattern constraints (guillotine and non-guillotine patterns), orientation of the pieces constraints, associated costs to the pieces constraints and demanding types of pieces constraints.

The second problem that we work on is the Two-Dimensional Single Bin Size Packing Problem, taking into account cutting pattern constraints (only guillotine patterns) and orientation pieces constraints. The last problem is the Container Loading Problem (Three-Dimensional Single Large Object Placement Problem) taking into account: orientation box constraints, load-bearing strength constraints, cargo stability constraints (full support) and multi-drop constraints.

All the previous problems have a big spectrum of application on the Industry, because of this; there is a big amount of previous work on it. Different methodologies, exact and approximate algorithms have been proposed as solution strategies. Due to the mathematical and computational complexity of these problems, the exact algorithms cannot solve real world instances of the problem. The approach of this study consists on presenting and/or adapting different encodings and optimization algorithms.

Among the proposed approach solutions is presented: a hybrid algorithm of optimization particle swarm, variable neighborhood descent and genetic algorithms to solve the knapsack and the bin packing problem and also a GRASP algorithm for the container loading problem.

Key-words - Bin Packing Problem. Container Loading Problem. Cutting and Packing Problems. Knapsack Problem. Metaheuristics.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Os problemas de corte e empacotamento são problemas clássicos dentro da pesquisa operacional e têm um grande espectro de aplicação na indústria. Neste contexto, o estudo tem por objetivo analisar três problemas diferentes que serão apresentados ao longo deste trabalho. Estes problemas são basicamente dois de corte e um de empacotamento.

1.1 DEFINIÇÃO DOS PROBLEMAS

Nos problemas de corte e empacotamento se definem dois elementos primordiais: as peças e os objetos. Este trabalho se limita ao estudo de peças e objetos que apresentam formas regulares. Nos problemas bidimensionais as peças (também chamadas de itens) se definem como pequenos retângulos, os quais devem ser localizados sobre objetos retangulares de um tamanho maior (também chamados de chapas, lâminas ou tabuleiros). Por outro lado, nos problemas tridimensionais as peças se definem como pequenos paralelepípedos (chamadas neste documento de caixas), as quais são localizadas dentro de um objeto que é um paralelepípedo de tamanho maior (chamado por comodidade de contêiner). A forma (ou localização) em que são cortadas ou empacotadas as peças sobre os objetos, constitui o que denominaremos *padrão*.

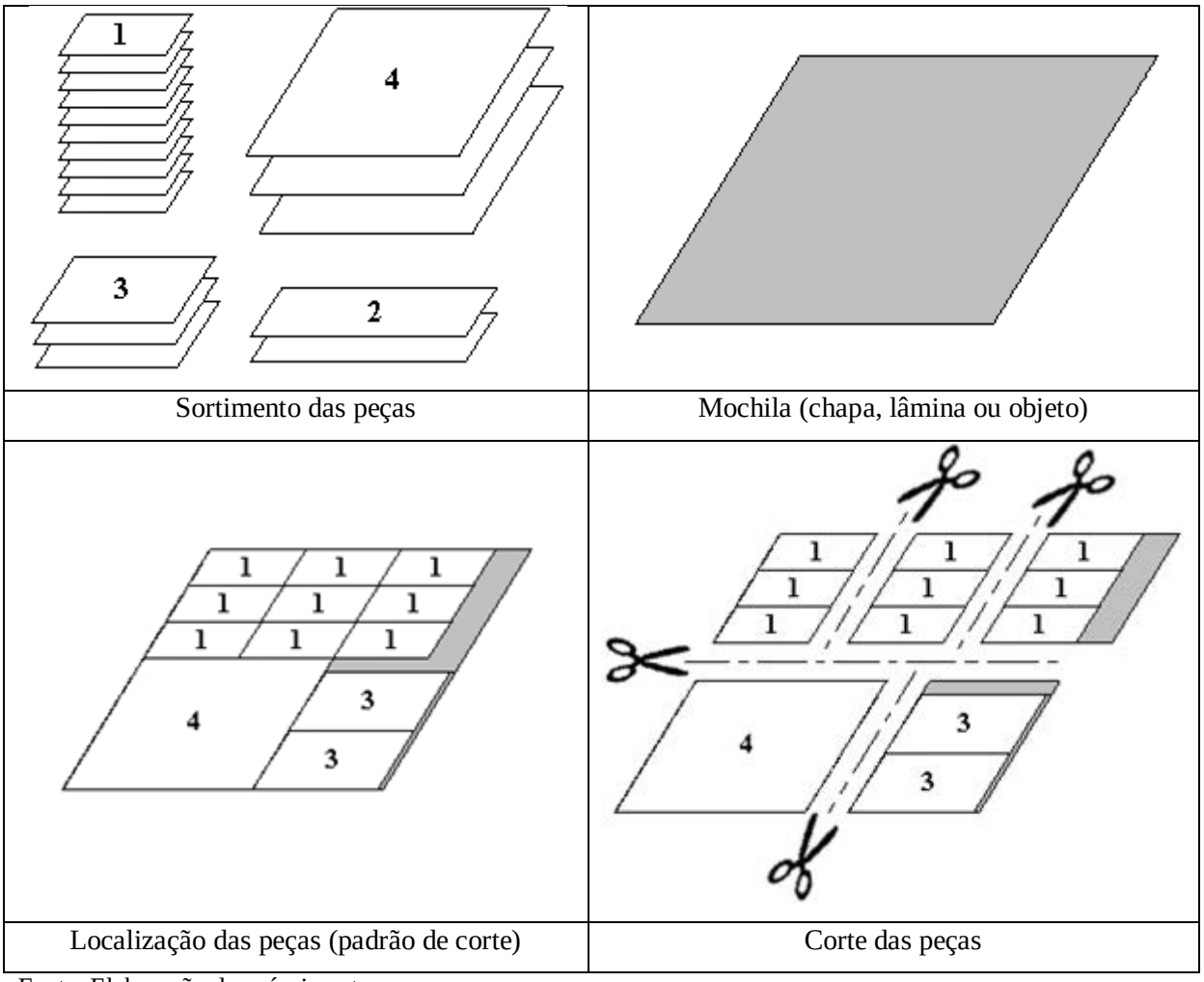
Na prática os problemas de corte e empacotamento são intuitivamente diferenciáveis, mas como sabemos estes problemas almejam alcançar o padrão de corte ou empacotamento mais eficiente que otimiza uma função objetivo; seja para maximizar o valor das peças cortadas (ou empacotadas), ou minimizar o número de matéria prima ou objetos necessários para o corte ou empacotamento das peças. Matemática e conceitualmente os padrões de corte e empacotamento são indistinguíveis, neste caso, são equivalentes. Isto significa que existe uma dualidade (correspondência) entre eles. A dualidade entre o material e o espaço que este ocupa (a ação de cortar material pode ser vista como empacotar o espaço que este ocupa e vice-versa, cortar o espaço que o material ocupa é o mesmo que empacotar o material).

Os problemas de corte e empacotamento são de interesse tanto no mundo acadêmico como no industrial, avançar nas pesquisas que abrangem esta temática de corte e empacotamento ótimo beneficia: empresas de corte, transporte, despacho e armazenamento de materiais (ou mercadorias), dado que representa uma estratégia de melhoramento importante, permitindo que estas possam maximizar o uso das matérias primas, fator que incide diretamente no custo

final do produto. Neste cenário se pode citar as indústrias de: corte de madeira, papel, metal, tecido e vidro, e as indústrias do transporte e armazenamento de caixas e paletes dentro de caminhões ou contêineres, que são as mais beneficiadas neste tipo de estudo.

O primeiro problema proposto pelo estudo, é o problema da mochila bidimensional (2D-SLOPP), este consiste em: cortar de um único objeto retangular chamado *mochila* um conjunto de pequenos retângulos chamados itens. O objetivo do problema é encontrar a localização das peças na mochila para ser cortadas de tal maneira que seja minimizado o espaço desperdiçado da chapa, sem sobrepor as peças entre si, e sem ultrapassar os limites da chapa. A este problema, se tem adicionado restrições práticas que representem cenários da vida real na indústria como são: restrições do padrão de corte (guilhotina e não guilhotina), restrições do valor associado às peças (ponderado ou não ponderado), restrições dos limites de exemplares das peças (restritos ou irrestritos) e restrições de orientação das peças (ver Figura 1).

Figura 1 - Problema da mochila bidimensional.

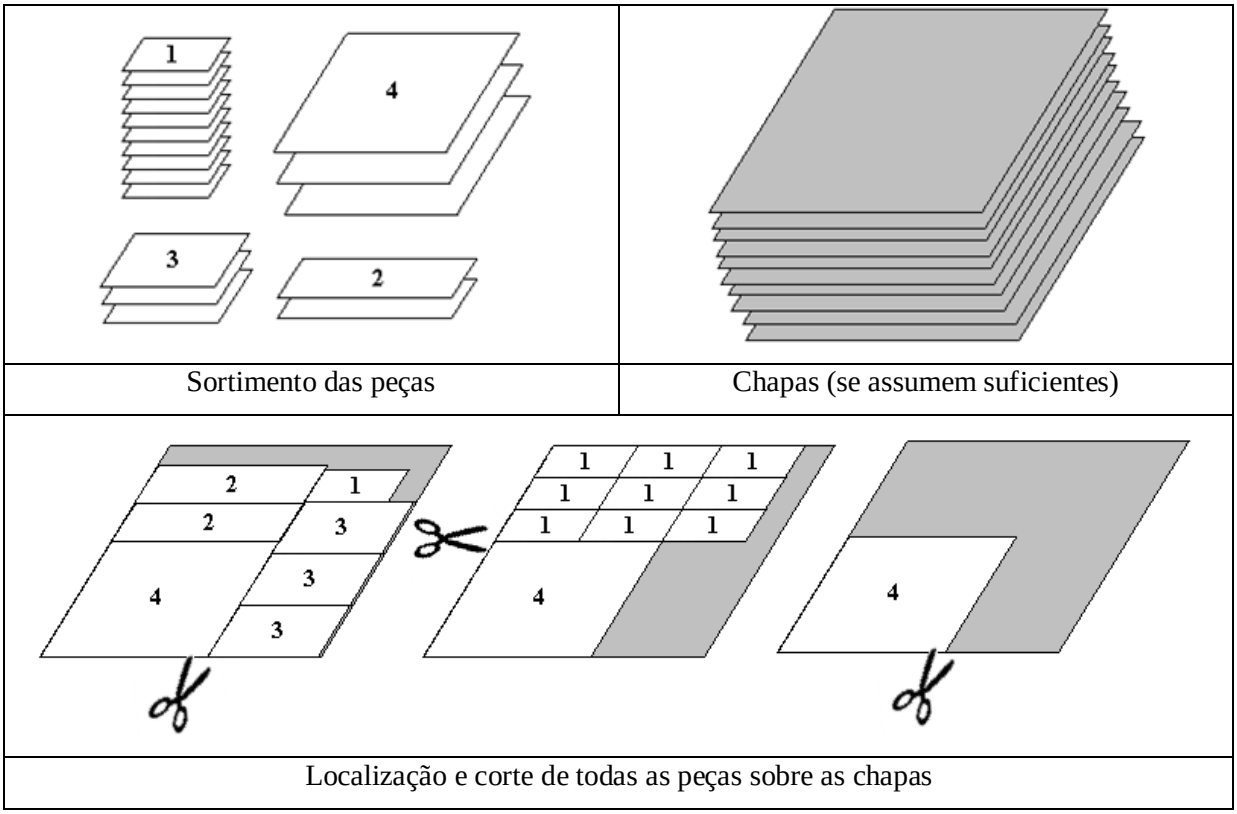


Sortimento das peças	Mochila (chapa, lâmina ou objeto)
Localização das peças (padrão de corte)	Corte das peças

Fonte: Elaboração do próprio autor

O problema da embalagem é uma generalização do problema da mochila. Neste problema existe um número suficiente de chapas para cortar a totalidade das peças (no problema da mochila tem-se só uma chapa). E este problema este consiste em cortar sobre um conjunto de objetos retangulares tudo um conjunto de pequenos retângulos. O objetivo encontrar a localização das peças sobre as chapas para serem cortadas, minimizando o número total de chapas, sem sobrepor as peças entre si e sem ultrapassar os limites das chapas. A este problema, se tem adicionado restrições práticas que representem cenários da vida real na indústria como são: restrições de padrões de corte tipo guilhotina e restrições de orientação das peças (ver Figura 2).

Figura 2 - Problema da embalagem.



Fonte: Elaboração do próprio autor

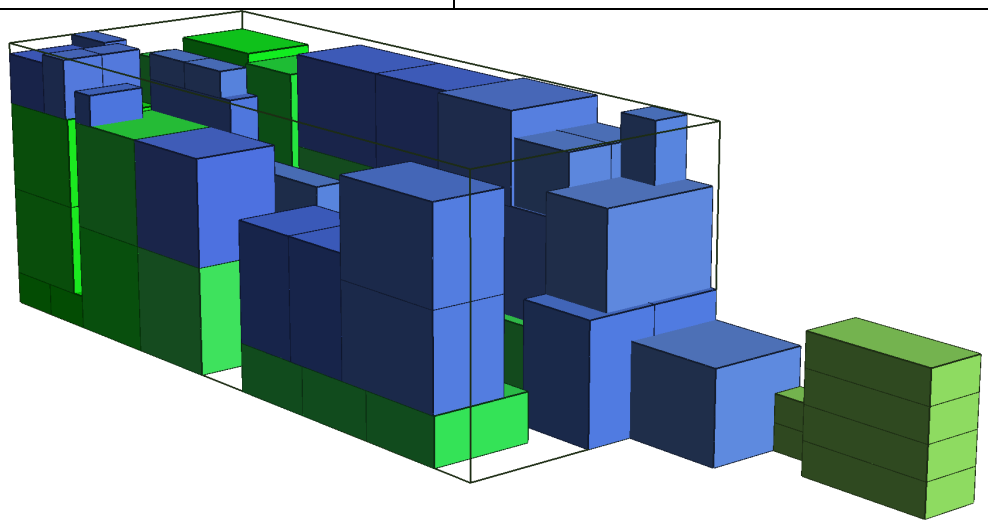
O terceiro e último problema de estudo, é o problema de carregamento do contêiner (3D-SKP ou 3D-SLOPP), que consiste em localizar dentro de um contêiner, um conjunto de caixas, tendo como objetivo maximizar o volume ocupado pelas peças carregadas, sem sobrepor as caixas entre si, e sem ultrapassar os limites do contêiner. A este problema, se tem adicionado restrições práticas que representem cenários da vida real na indústria, como são restrições de: orientação das caixas, limites resistência de peso das caixas ao empilhamento, limite de peso

do carregamento suportado pelo contêiner, estabilidade do carregamento e carregamento fracionado em múltiplos destinos (ver Figura 3).

Figura 3 - Problema de carregamento do contêiner.

Sortimento das caixas

Contêiner



Localização das caixas dentro do contêiner e das caixas restantes que tem ficado por fora (Padrão de carregamento)

Fonte: Elaboração do próprio autor

1.2 MOTIVAÇÃO

Este estudo tem por base propor uma revisão do estado da arte sobre os problemas de corte e empacotamento, que podem ser utilizados como referência de base teórica para a comunidade acadêmica, no intuito de expor representações e metodologias diferenciadas que solucionem os problemas de: 2D-SLOPP, 2D-SBSBPP e 3D-SLOPP. Estes três problemas têm sido demonstrados serem *NP-Hard* por Garey e Johnson (1979) e Beasley (2004).

Existe uma grande quantidade de estudos formais sobre os problemas de corte e empacotamento, livros como: Brown (1971), Martello e Toth (1990), Dyckhoff e Finke (1992), e Dyckhoff *et al.* (1997). Classificações e tipologias como: Hinxman (1980), Dyckhoff (1990), Wäscher *et al.* (2007) e Bortfeldt e Wäscher (2013). E artigos de revisão como: Golden (1976), Dowsland (1985), Dyckhoff *et al.* (1985), Haessler e Sweeney (1991), Dowsland e Dowsland (1992), Morabito e Arenales (1992), Ram (1992), Sweeney e Paternoster (1992), Cheng *et al.* (1994), Martello (1994a; 1994b), Bischoff e Wäscher (1995), Coffman *et al.* (1996), Arenales *et al.* (1999), Hopper e Turton (2001a; 2001b), Hifi (2002b), Lodi *et al.* (2002a), Wang e Wäscher (2002), Oliveira e Wäscher (2007). Nestes trabalhos se discriminam as diferentes carências e se apontam os novos horizontes na pesquisa sobre os problemas de corte e empacotamento.

Sob esta perspectiva, cabe ressaltar que nas duas últimas décadas no cenário acadêmico e principalmente na indústria, se tem observado um grande interesse sobre as pesquisas desta natureza. Em especial no que tange ao desenvolvimento nas áreas de pesquisa operacional e engenharia da produção, crescendo assim o interesse pelas metodologias que apresentem soluções para os problemas de corte e empacotamento, objetivando desta forma melhorar o desempenho tanto em qualidade, como em tempo computacional de resposta, e incorporação de restrições práticas que representem situações reais na indústria.

1.3 OBJETIVOS

1.3.1 Objetivo geral

O objetivo deste trabalho é desenvolver uma metodologia de solução aproximada para os problemas da mochila bidimensional, o problema da embalagem e o problema do carregamento do contêiner, incluindo restrições práticas dos cenários reais dos problemas.

1.3.2 Objetivos específicos

Revisar o estado da arte dos modelos matemáticos que formulem os problemas de interesse neste estudo incluindo as restrições práticas desejadas.

Revisar o estado da arte das metodologias de solução propostas para este tipo de problemas.

Utilizar ou propor representações que permitam a implementação de técnicas heurísticas e meta-heurísticas de otimização combinatória, para resolver os problemas de corte e empacotamento que fazem parte do universo deste estudo.

Implementar ou desenvolver metodologias de solução para os problemas propostos neste estudo.

Validar as metodologias de solução desenvolvidas utilizando as diferentes instâncias de casos teste apresentados na literatura especializada.

1.4 ALCANCE E CONTRIBUIÇÕES

O alcance deste trabalho consiste em desenvolver diferentes metodologias de solução para os problemas de: a mochila bidimensional, o problema da embalagem e o problema do carregamento do contêiner. Onde o desempenho e a qualidade do estudo realizado, serão respaldados pelos resultados obtidos pelo conjunto de metodologias apresentadas na literatura. As implementações realizadas neste trabalho devem ser comparadas com as metodologias representativas do estado da arte de otimização exata e aproximada.

A fim de conhecer a sensibilidade dos algoritmos propostos e a qualidade de suas respostas, se efetua uma análise dos resultados obtidos que nos permita classificar o conjunto de técnicas propostas de acordo a seu desempenho.

Dentro das metodologias propostas se encontram contribuições como:

- O uso de uma codificação de árvores de cortes para os problemas de corte bidimensionais.
- O desenvolvimento de uma heurística de otimização especializada para os problemas de corte. Baseada em otimização por enxame de partículas combinado com busca em vizinhança variável e algoritmos genéticos.
- A esquematização de uma única metodologia para dar soluções a diferentes problemas de corte e empacotamento da mesma família.
- A implementação e adaptação do algoritmo GRASP (do inglês, *Greedy Randomized Adaptive Search Procedure*) para solucionar o problema do carregamento de contêineres com restrições práticas de empacotamento.

1.5 PUBLICAÇÕES CIENTÍFICAS RELACIONADAS COM A TESE

Os capítulos apresentados nesta tese são baseados em artigos já publicados ou submetidos à avaliação tanto em jornais indexadas como em anais de congressos.

O desenho e desenvolvimento da representação de árvores de cortes e o algoritmo de otimização por enxame de partículas, para o problema da mochila com restrições de corte tipo guilhotina descrito no capítulo 3, está baseado nos seguintes artigos:

- D. Álvarez, E. M. Toro, R. A. Gallego, (2010), Problema de la mochila irrestricta bidimensional guillotinada, *Revista Ingeniería & Universidad*, Universidad Javeriana, Bogotá (Colombia), 14 (2): 327-344, julio-diciembre. ISSN 0123-2126.
- D. Álvarez, L. M. Escobar, R. A. Gallego, (2011), Unconstrained k-staged two-dimensional guillotineable single knapsack problem, *In Proceedings do 43th Simpósio Brasileiro de Pesquisa Operacional*, Ubatuba (Brasil), 43: 95-107, Agosto. ISSN 1518-1731.
- D. Álvarez, R. A. Romero, (2012), Particle swarm optimization with turbulence factor for two-dimensional guillotine single knapsack problems, *In Proceedings do 44th Simpósio Brasileiro de Pesquisa Operacional/16th Congreso Latino-Iberoamericano de Investigación Operativa*, Rio de Janeiro (Brasil), 44: 204-215, Setembro. ISSN 1518-1731.

A proposta e o desenvolvimento da representação de árvores de cortes e o time assíncrono de meta-heurísticas para o problema da mochila com restrições de corte tipo não guilhotina descrito no capítulo 3, está baseado nos seguintes dois artigos:

- D. Álvarez, L. M. Escobar, R. A. Romero, (2011), Equipo asíncrono de agentes basados en recocido simulado aplicado al problema del agente viajero simétrico, *Revista Scientia et Technica*, Universidad Tecnológica de Pereira, Pereira (Colombia), 49: 122-127, Diciembre. ISSN 0122-1701.
- D. Álvarez, L. M. Escobar, R. A. Romero, (2013), A hybrid algorithm of Particle Swarm Optimization and Genetic Algorithms for the non-guillotine two-dimensional single knapsack problem, *Presented on the 26th European Conference on Operational Research and Submitted for publication on a journal.*

A proposta e o desenvolvimento da representação em árvores de cortes e os algoritmos de otimização por enxame de partículas para o problema da embalagem descrito no capítulo 6, é baseado no artigo:

- D. Álvarez, E. M. Toro, R. A. Gallego, (2011), Estudio comparativo de algoritmos basados en cúmulo de partículas para resolver el problema de empaquetamiento en placas, Revista Ingeniería y Competitividad, Universidad del Valle, Cali (Colombia), 13 (1): 113-130, Mayo. ISSN 0123-3033.

A proposta e o desenvolvimento da representação dos espaços máximos e o algoritmo GRASP para o problema do carregamento do contêiner descrito no capítulo 4, é baseado no artigo:

- D. Álvarez, F. Parreño, R. Álvarez-Valdés (2013), A heuristic approach to the Container Loading Problem with multi-drop constraints, *Presented on the 26th European Conference on Operational Research and Submitted for publication on a journal.*

1.6 ORGANIZAÇÃO DO DOCUMENTO

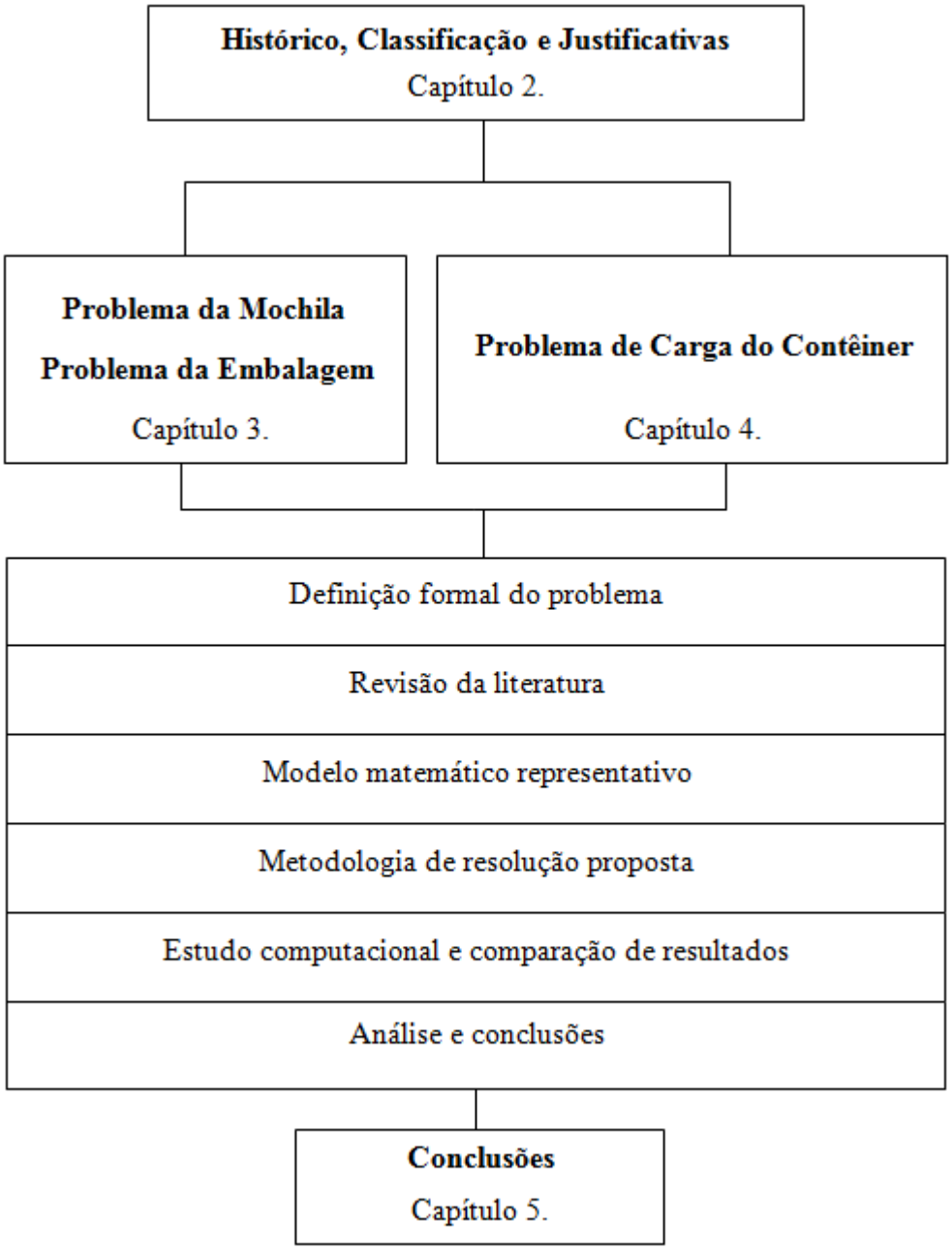
O resto do texto está organizado da seguinte maneira (ver Figura 4): no Capítulo 2 apresenta-se um resumo dos históricos dos problemas de corte e empacotamento e se detalha sua classificação e tipologia, dando assim uma revisão inicial para o entendimento completo dos problemas trabalhados neste estudo, no final deste capítulo justifica-se o estudos dos problemas.

No Capítulo 3 é definido formalmente o problema da mochila e cada uma das restrições consideradas. Apresenta-se então uma revisão das diferentes metodologias propostas na literatura, cobrindo assim: métodos exatos, heurísticas e meta-heurísticas. Além disto, é apresentada a formulação matemática representativa do problema. Apresenta-se a metodologia de solução proposta, acompanhada de um estudo computacional e a comparação de resultados obtidos com respeito aos da literatura. Realiza-se uma análise e se apresentam as conclusões. Por outro lado, este mesmo esquema de investigação e solução tem sido usado sobre o problema da embalagem e é apresentado na ultima seção deste capítulo.

No Capítulo 4 discutimos sobre o problema de carregamento do contêiner na perspectiva de realizar os mesmos passos para por fim, apresentar a metodologia de solução.

Finalmente, no Capítulo 5 conclui-se esta tese, resumindo as principais contribuições alcançadas neste estudo, seguido dos possíveis horizontes de pesquisa sobre esta temática.

Figura 4 - Estrutura da tese.



Fonte: Elaboração do próprio autor

2 HISTÓRICO, CLASSIFICAÇÃO E REVISÃO DA LITERATURA.

Desde a publicação de Gilmore e Gomory (1961; 1963) sobre os problemas de corte e empacotamento, o número de pesquisadores e pesquisas no mundo inteiro sobre esta temática tem aumentado a cada década. Devido a quantidade de trabalhos publicados na literatura especializada tem sido difícil ordenar, classificar e ponderar estes para uma conciliação dos problemas futuros com maior importância.

A primeira tentativa de classificar os problemas de corte e empacotamento foi proposta pelo Dyckhoff (1990), tempo depois uma tipologia melhorada foi apresentada pelo Wäscher et al. (2007). E devido ao interesse ganhado pelo problema de carregamento de contêineres, Bortfeldt e Wäscher (2013) apresentam uma atualização desta tipologia.

Neste documento se apresenta a tipologia de Wäscher et al. (2007) para de esclarecer pontualmente os problemas abordados neste estudo e justificar a escolha destes para estudo.

2.1 CLASSIFICAÇÃO DE WÄSCHER ET AL. (2007)

Lembrando a estrutura dos problemas de corte e empacotamento, estes são compostos por dois conjuntos de elementos: um conjunto de grandes objetos (entradas ou suprimentos) e um conjunto de pequenas peças (saídas ou demanda), os quais são definidos geometricamente usando um número de dimensões necessárias. Assim, o problema consiste em selecionar algumas ou a totalidade das pequenas peças, para atribuí-las e localizá-las dentro de um conjunto dos grandes objetos, de forma tal que se conservem as seguintes restrições geométricas: cada pequena peça deve encontrar-se totalmente dentro do objeto e as pequenas peças não devem se sobrepor entre si. A forma ou localização em que se encontram as peças sobre os objetos é denominada como padrão. Encontrar o padrão de corte ou empacotamento mais eficiente com relação a uma ou várias funções objetivos determinadas resume o objetivo dos problemas de corte e empacotamento.

Formalmente, cinco subproblemas podem ser identificados, dentre os quais devem ser resolvidos simultaneamente para alcançar a solução ótima de um problema de corte ou empacotamento:

• Selecionar os grandes objetos;

• Selecionar as pequenas peças;

• Agrupar as pequenas peças;

• Atribuir os subconjuntos de peças nos grandes objetos;

• Localizar as peças em cada um dos objetos selecionados conservando as restrições geométricas.

Tipos especiais dos problemas de corte e empacotamento podem ser caracterizados por propriedades adicionais. Em particular, estes podem degenerar-se no sentido que talvez não abranjam os subproblemas anteriormente mencionados.

Wäscher et al. (2007) utiliza cinco critérios para realizar sua classificação dos problemas de corte e empacotamento:

- Dimensionalidade: Representa o número das dimensões geométricas necessárias e relevantes para descrever as peças e os objetos.
- Classe de atribuição: Representa o objetivo conceitual, tendo dois possíveis: a maximização da saída (*output maximization*) e a minimização da entrada (*input minimization*).

  Na maximização da saída, um conjunto das peças tem que ser atribuído a um conjunto dado de objetos. Porém, de antemão se sabe que talvez nem usando em sua totalidade o conjunto de objetos grandes seja suficiente para acomodar todas as peças. Portanto, o objetivo é encontrar a localização das peças sobre os objetos, que maximize o beneficio das peças atribuídas.

  Na minimização da entrada, a totalidade das peças deve ser atribuída ao conjunto de objetos grandes. Ao contrário do anterior, o conjunto de objetos é suficiente para acomodar todas as peças. Portanto, o objetivo é encontrar a localização das peças sobre os objetos que minimize o valor dos objetos atribuídos.
- Sortimento das peças: Representa os quão variados ou diversos são as peças com relação as suas formas e medidas, mas levando em conta somente as dimensões geométricas relevantes. As peças podem ser agrupadas em relativamente (com relação ao número total de peças) poucas classes, nas quais, as peças têm idênticas formas e dimensões. Por definição, se tratam como diferentes classes (ou tipos como depois serão mencionadas neste documento) de peças, se estas têm as mesmas formas e

dimensões, mas suas orientações são diferentes. Para isto ocorrem três casos distintos: Peças idênticas, Sortimento fracamente heterogêneo e Sortimento fortemente heterogêneo.

Um sortimento de peças idênticas se apresenta quando, todas as peças têm tanto formas e dimensões iguais, ou seja, só existe uma classe (tipo) de peça. Em casos onde o objetivo é a maximização da saída, assume-se que o único tipo de peça demandado tem um número ilimitado de copias ou exemplares.

Um sortimento fracamente heterogêneo se apresenta quando, a demanda de cada tipo de peça é relativamente grande, ou seja, existe um número alto de copias ou exemplares por tipo de peça, e este pode ser ou não restringido por um limite máximo.

Um sortimento fortemente heterogêneo se apresenta quando, o número de exemplares de cada tipo de peça é relativamente baixo, sendo quase um exemplar só por classe de peça.

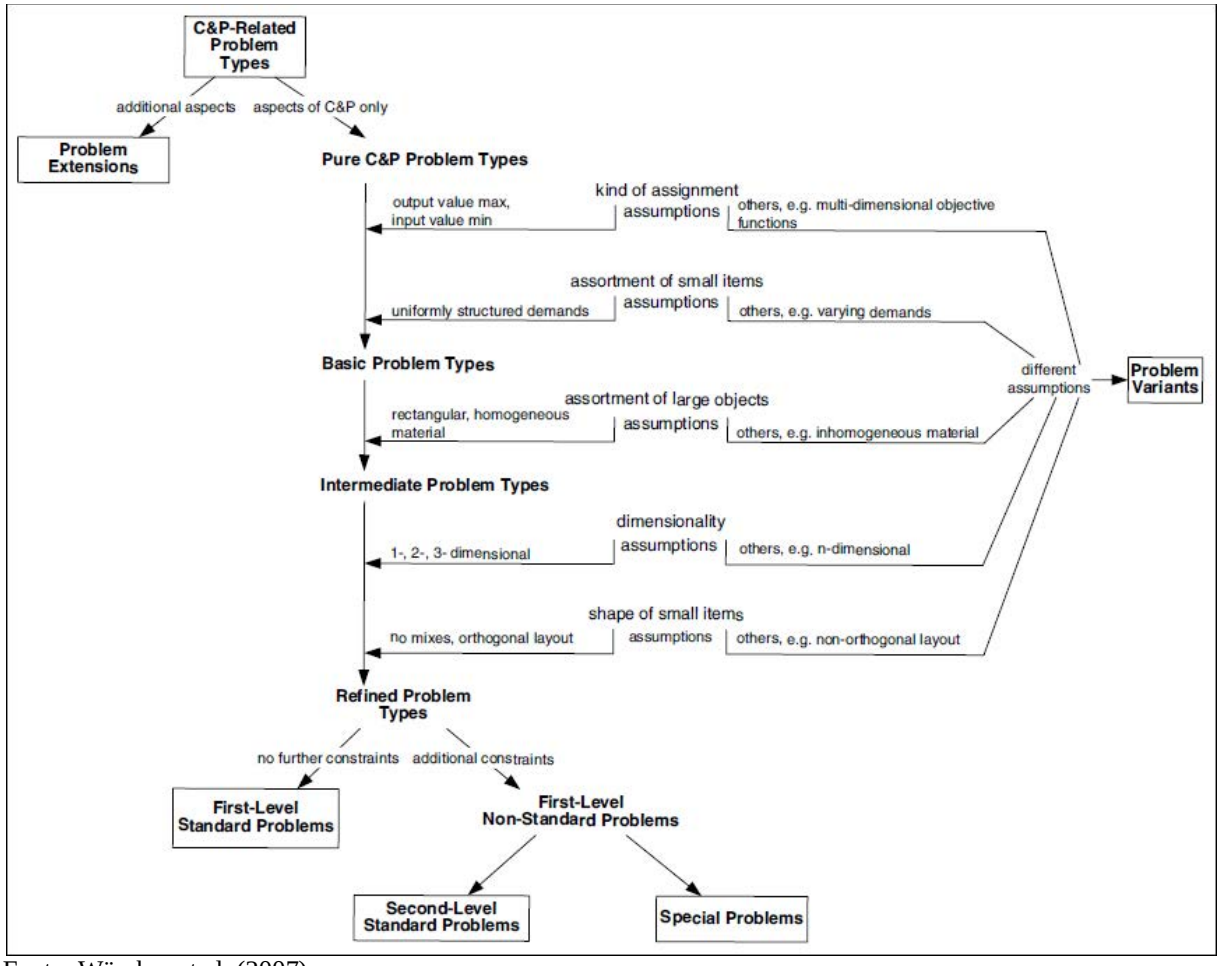
- Sortimento dos objetos: Representa o quão diversos são os objetos com relação as suas formas e dimensões, em especial, algumas ou todas as dimensões podem ser fixas (dadas) ou variáveis (ilimitadas). Para isto se distinguem dois casos distintos: existe somente um objeto ou existem vários objetos.

  Quando existe só um objeto pode ocorrer que este tenha todas suas dimensões fixas (sejam dadas) ou que uma ou mais de suas dimensões sejam variáveis ou ilimitadas. Quando existem vários objetos cada um destes tem suas dimensões fixas, assim não existe o caso de vários objetos com dimensões variáveis. Deste modo, a existência de vários objetos significa que, igualmente às peças, estes possam ser classificados novamente em três possíveis categorias de sortimento: que todos os objetos sejam idênticos, que seu sortimento seja fracamente heterogêneo ou que seu sortimento seja fortemente heterogêneo.
- Forma das peças: representa geometricamente como estão definidas as peças, nos casos aonde são relevantes e necessárias duas ou três dimensões geométricas para representar as peças, podem-se identificar duas classes de peças: As regulares (retângulos, círculos, paralelepípedos, cilindros, bolas, etc.) e as irregulares. Neste trabalho limitamo-nos ao estudo de problemas onde as formas das peças são regulares e em especial suas formas são retangulares, o que limita os problemas a trabalharem com retângulos e paralelepípedos, para os problemas de duas e três dimensões respectivamente. Além disto, neste trabalho os objetos também devem ter forma

retangular. Por outro lado, se assume que todas as localizações das peças sobre os objetos se realizam ortogonalmente, ou seja, no momento de colocar uma peça seus lados devem ser colocados paralelos aos lados do objeto.

Com estes cinco critérios pode-se definir um esquema dos tipos de problemas de corte e empacotamento. A Figura 5 ilustra a diferença entre tipos de problemas "puros" de corte e empacotamento, e extensões do problema, os quais se diferenciam por incluir aspectos adicionais que não fazem parte das características inerentes do corte e empacotamento.

Figura 5 - Resumo dos tipos de problemas de corte e empacotamento



Fonte: Wäscher et al. (2007).

Dentro dos tipos de problemas puros de corte e empacotamento vemos que eles podem ser divididos entre: tipos básicos, tipos intermediários e tipos refinados do problema, a sua vez uma hipótese distinta das clássicas, em qualquer critério de classificação converte o problema em uma "variante" do problema. Por outro lado, os casos refinados depois são definidos ao aplicar outros critérios sobre os tipos básicos do problema.

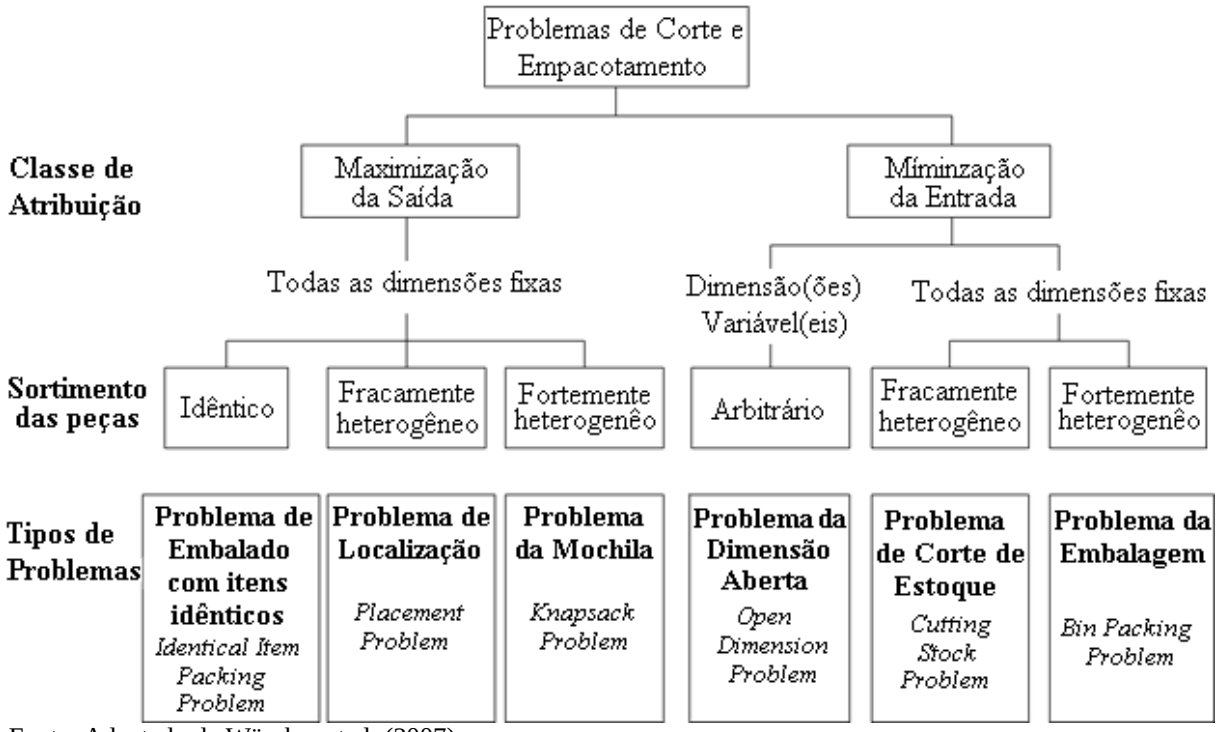
É importante detalhar os tipos básicos dos problemas de corte e empacotamento, onde se definirão os tipos refinados, sendo estes últimos a categoria onde estão enquadrados os problemas abordados neste estudo. Para isto, é necessário aprofundar-nos nos critérios de classe de atribuição e sortimento das peças.

Os problemas onde o objetivo é a maximização da saída têm em comum que os objetos são fornecidos em quantidades limitadas e não são suficientes para acomodar todas as peças. Como o valor das peças localizadas deve ser maximizado, todos os objetos terminarão sendo usados. Em outras palavras, geralmente há um problema de seleção das peças, mas não um dos objetos.

Por outro lado, nos problemas onde o objetivo é a minimização da entrada, têm como característica o fato de que o fornecimento de objetos é o suficientemente grande para acomodar todas as peças. A demanda das peças deve ser totalmente satisfeita, isto significa que não existe problema de seleção das peças. Desta forma, o valor dos objetos necessários para localizar todas as peças deve ser minimizado.

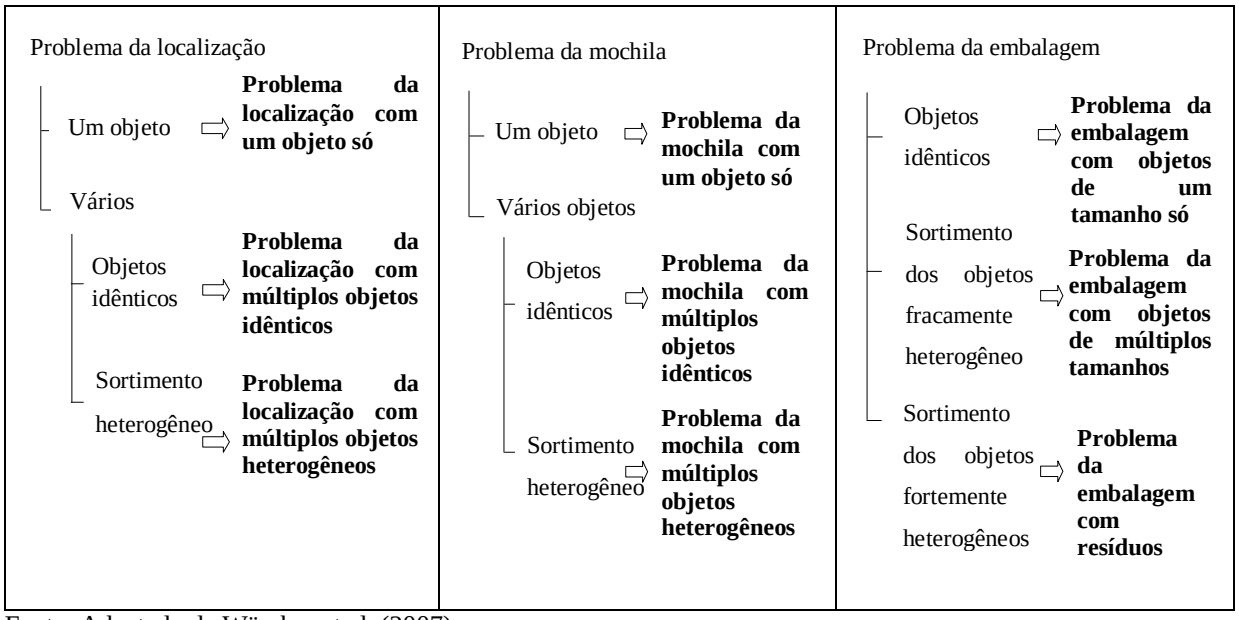
Na Figura 6 ilustram-se os tipos básicos de problemas.

Figura 6 - Tipos básicos dos problemas de corte e empacotamento



Fonte: Adaptado de Wäscher et al. (2007)

Figura 7 - Tipos Intermediários de problemas (Problema da Localização, Problema da Mochila e Problema da Embalagem)



Fonte: Adaptado de Wäscher et al. (2007)

- **Problema de acondicionamento de itens idênticos**

Esta categoria de problemas consiste em atribuir o maior número possível de peças idênticas a um conjunto dado de objetos. Nota-se que, devido ao fato de que todas as peças serem idênticas, realmente não existe um problema de selecionar e agrupar as peças, nem existe um verdadeiro problema de atribuição das peças aos objetos. Em outras palavras, a estrutura geral anteriormente mencionada, se reduz ao problema de localizar as peças a cada um dos objetos cumprindo suas restrições geométricas.

- **Problema da localização**

Na literatura, os problemas desta categoria são conhecidos com diferentes nomes. Neste trabalho se usa o termo de Problema da localização, para definir a categoria de problemas nas quais um sortimento de peças fracamente heterogêneo deve ser atribuído a um conjunto limitado de objetos. O valor ou o tamanho total (como um objetivo auxiliar) das peças localizadas deve ser maximizado, ou alternativamente, as respectivas perdas devem ser minimizadas.

- **Problema da mochila**

Esta categoria de problemas é caracterizada por um sortimento de peças fortemente heterogêneo, o qual deve ser colocado num conjunto de objetos. A disponibilidade de objetos

é limitada, de forma tal que não é garantido que todas as peças possam ser acomodadas. Portanto, o valor das peças localizadas deve ser maximizado.

Foram enunciados anteriormente os três problemas básicos que fazem parte dos tipos de problemas, onde o objetivo é a maximização da saída. Por outro lado, os tipos de problemas onde o objetivo é a minimização da entrada, se caracterizam porque o fornecimento de objetos é suficientemente grande como para acomodar todas as peças. O conjunto de peças demandadas deve ser satisfeito em sua totalidade, porque não existe o problema de selecionar as peças. Por enquanto, o valor dos objetos necessários para acomodar todas as peças deve ser minimizado. Dentro desta categoria se encontram os seguintes problemas:

- **Problema de dimensão aberta**

Esta categoria de problemas consiste em acomodar a totalidade do conjunto de peças sobre um ou mais objetos. Os objetos são determinados, porém sua extensão em ao menos uma dimensão pode ser considerada variável (indefinida ou ilimitada). Em outras palavras, este problema implicará a decisão de fixar a dimensão variável do objeto. Por outro lado, só o conjunto de objetos necessários para acomodar todas as peças representam as “entradas” no sentido geral da estrutura dos problemas de corte e empacotamento. Isto significa que o valor da entrada (ou uma medida auxiliar como a extensão, o tamanho ou o volume) é o que deve ser minimizado.

- **Problema de corte de estoque**

Esta categoria de problemas exige que um sortimento de peças fracamente heterogêneo deva ser localizado na sua totalidade numa seleção de objetos de mínimo valor, número ou tamanho total. Ao contrário do problema anterior, a extensão dos objetos é dada (fixada) em todas suas dimensões geométricas. Destaca-se que não se faz nenhuma hipótese a respeito do sortimento dos objetos, que poderia ser como já sabemos: um sortimento de objetos idênticos, um sortimento de objetos fracamente heterogêneo ou um sortimento de objetos fortemente heterogêneo.

- **Problema da embalagem**

Esta categoria de problemas é caracterizada por um sortimento de peças fortemente heterogêneo. De novo, as peças devem ser atribuídas a um conjunto de objetos idênticos, ou um sortimento de objetos fraca ou fortemente heterogêneo. Portanto, o valor, o número ou o tamanho total (ou outro objetivo auxiliar) dos objetos necessários, deve ser minimizado.

Figura 8 - Panorama dos tipos intermediários do problema; maximização da saída

Características dos objetos \ Sortimento das peças		Idênticas	Fracamente Heterogêneas	Fortemente Heterogêneas
Todas as dimensões fixas	Um único objeto	**Problema de acondicionamento de itens idênticos** *Identical Item Packing Problem* IIPP	**Problema da localização num único objeto** *Single Large Object Placement Problem* SLOPP	**Problema de uma única mochila** *Single Knapsack Problem* SKP
	Objetos idênticos		**Problema da localização em múltiplos objetos idênticos** *Multiple Identical Large Object Placement Problem* MILOPP	**Problema das múltiplas mochilas idênticas** *Multiple Identical Knapsack Problem* MIKP
	Objetos Heterogêneos		**Problema da localização em múltiplos objetos heterogêneos** *Multiple Heterogeneous Large Object Placement Problem* MILOPP	**Problema das múltiplas mochilas heterogêneas** *Multiple Heterogeneous Knapsack Problem* MIKP

Fonte: Adaptado de Wäscher et al. (2007)

Os seis problemas anteriores, compões os tipos básicos de problemas de corte e empacotamento (ver Figura 6). Porém, para esclarecer e detalhar os problemas considerados neste estudo é necessário continuar com a estrutura de classificação, passando assim aos tipos intermediários do problema. Portanto, continuaremos ao adicionar o critério de sortimento dos objetos. Concentrando-nos agora unicamente em três tipos básicos de problemas: o problema da localização, o problema da mochila e o problema da embalagem (ver Figura 7).

No problema da localização pode-se considerar que existem um ou mais objetos. Se só existe um único objeto, isto indica que não existe o problema de selecionar objetos, pois é o único objeto que faz parte da entrada, portanto sempre será atribuído. Esta categoria de problemas é chamada de problema da localização num único objeto. Por outro lado, se existe mais de um objeto, implicará a inclusão do problema de selecionar os objetos que podem ter um sortimento de objetos idênticos ou um sortimento de objetos heterogêneos. Estas categorias de problemas são chamadas como problema da localização em múltiplos objetos idênticos e problema da localização em múltiplos objetos heterogêneos.

Figura 9 - Panorama dos tipos intermediários do problema; minimização da entrada

<table>
<tr><td colspan="2">Sortimento das peças
Características de os objetos</td><td>Fracamente Heterogêneas</td><td>Fortemente Heterogêneas</td></tr>
<tr><td rowspan="3">Todas as dimensões fixas</td><td>Objetos idênticos</td><td>Problema de corte de estoque em objetos de tamanhos idênticos
Single Stock Size
Cutting Stock Problem
SSSCSP</td><td>Problema da embalagem em objetos de um único tamanho
Single Bin Size
Bin Packing Problem
SBSBPP</td></tr>
<tr><td>Objetos fracamente heterogêneos</td><td>Problema de corte de estoque múltiplos objetos
Multiple Stock Size
Cutting Stock Problem
MSSCSP</td><td>Problema da embalagem em objetos múltiplos tamanhos
Multiple Bin Size
Bin Packing Problem
MBSBPP</td></tr>
<tr><td>Objetos fortemente heterogêneos</td><td>Problema residual de corte de estoque
Residual
Cutting Stock Problem
RCSP</td><td>Problema residual da embalagem
Residual
Bin Packing Problem
RBPP</td></tr>
<tr><td colspan="2">Um único objeto com dimensões variáveis</td><td colspan="2">Problema de dimensão aberta
Open Dimension Problem
ODP</td></tr>
</table>

Fonte: Adaptado de Wäscher et al. (2007)

No problema da mochila podemos encontrar que existam uma ou mais mochilas (ou objetos). Uma vez mais, se só existe uma mochila isto representa que não existe o problema de selecionar objetos, pois é a única que faz parte da entrada, portanto sempre será atribuída.

Esta categoria de problemas é chamada de problema de uma única mochila. Por outro lado, se existe mais de uma mochila, implicará a inclusão do problema de selecionar os objetos, estes por sua vez podem ter um sortimento de mochilas idênticas ou um sortimento de mochilas heterogêneas. Estas categorias de problemas são chamadas de problema de múltiplas mochilas idênticas e problema de múltiplas mochilas heterogêneas, respectivamente.

No problema da embalagem podemos encontrar um ou mais tamanhos diferentes dos objetos. Se só existe um tamanho de objeto o problema é chamado de problema da embalagem em objetos de um único tamanho. Por outro lado, se existe mais de um tamanho de objetos, pode ser que o sortimento dos seus tamanhos seja fraco ou fortemente heterogêneo. Estas categorias de problemas são chamadas de problema da embalagem em objetos de múltiplas tamanhos e problema residual de embalagem, respectivamente.

Por outro lado, as Figuras 8 e 9 ilustram todos os tipos intermediários dos problemas de corte e empacotamento. Portanto, somente falta a definição dos tipos “refinados”. Estes são alcançados ao incluir os critérios da dimensionalidade e da forma das peças. Obtendo assim subcategorias que se caracterizam ao adicionar o adjetivo (dimensionalidade e forma) ao nome do tipo intermediário do problema.

Como ressaltado anteriormente, este trabalho se limita ao estudo de problemas onde as peças têm forma regular e o número de dimensões relevantes para descrever as peças e objetos são duas ou três dimensões. Deste modo, os tipos refinados dos problemas levados em conta neste estudo são:

- O problema bidimensional de uma única mochila retangular, 2D-SLOPP da sigla em inglês de *Two-Dimensional, rectangular Single Large Object Packing Problem*. Por facilidade pode ser denominado de Problema da Mochila.
- O problema da embalagem em objetos bidimensionais retangulares de um único tamanho, 2D-SBSBPP da sigla em inglês de *Two-Dimensional, rectangular Single Bin Size Bin Packing Problem*. Neste documento este problema é denominado de Problema da Embalagem.
- O problema de localização tridimensional em um único objeto retangular, 3D-SLOPP da sigla em inglês de *Three-Dimensional, rectangular Single Large Object Packing Problem*. Este problema é também conhecido como problema do carregamento do contêiner (em inglês *Single Container Loading (Packing) Problem*).

Estes três problemas selecionado têm um grande interesse devido a suas já bem conhecidas aplicações na indústria. Isto quer dizer que são problemas que devem ser resolvidos no dia a dia das empresas que trabalham em corte, empacotamento, transporte e armazenamento de mercadorias. Apesar de existir uma vasta quantidade de trabalhos sobre estes problemas na literatura, poucos trabalhos apresentam ou incluem em suas metodologias restrições práticas que representem situações reais dos problemas, ou seja, muitos estudos publicados se dedicam somente a resolver os tipos "refinados" padronizados (*First-Level Standard Problems*), ao contrário dos trabalhos anteriores, neste trabalho se pretende incluir o maior número de restrições práticas, porém conservando a estrutura geral dos problemas de corte e empacotamento, de forma tal que estes problemas não sejam classificados como situações especiais ou simples variantes do problema. Portanto, os problemas estudados neste trabalho pertencem aos tipos refinados padronizados de segundo nível (*Second-Level Standard Problems*). Em seguida será definido o conjunto de restrições adicionais levadas em conta neste estudo.

2.2 DESCRIÇÃO DAS RESTRIÇÕES ADICIONAIS

Como foi ressaltado anteriormente, três diferentes tipos de problemas serão estudados neste trabalho, além de suas versões padrões, é adicionado um conjunto de restrições práticas, encontradas em situações reais dos problemas.

2.2.1 Restrições adicionais ao problema da mochila

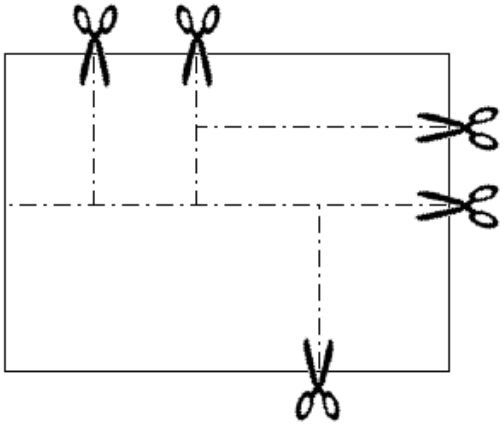
Conforme a classificação enunciada anteriormente, as restrições adicionais que serão consideradas podem ser divididas pelos critérios: padrões de corte e empacotamento, e as características das peças.

- Padrões de corte o empacotamento bidimensional

Morabito e Arenales (1996) classificam os padrões de empacotamento em padrões guilhotina e não guilhotina (ver Figura 20). Um corte tipo guilhotina é aquele que ao ser aplicado sobre um retângulo produz dois novos retângulos, ou seja, se o corte vai de um extremo ao outro do retângulo original; caso contrário se denomina do tipo não guilhotina. Um padrão do tipo guilhotina é possível obter-se por sucessivos cortes de tipo guilhotina (ver Figura 10), ou seja, os padrões estão condicionados pela tecnologia de corte que só tem um grau de liberdade (uma dimensão geométrica) que acostumamos chamar de guilhotina. Por outro lado, um

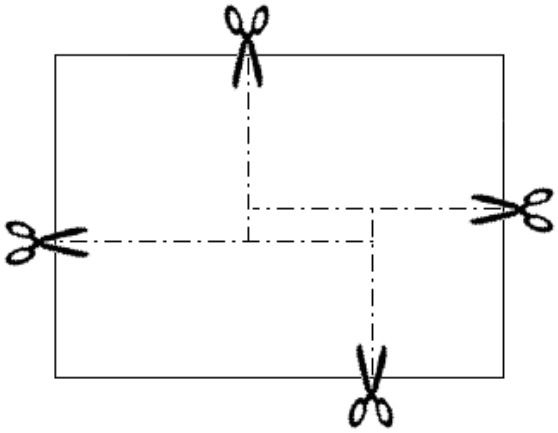
padrão é do tipo não guilhotina, se é obtido por sucessivos cortes de guilhotina e não guilhotina (ver Figura 11).

Figura 10 - Padrão guilhotina



Fonte: Elaboração do próprio autor

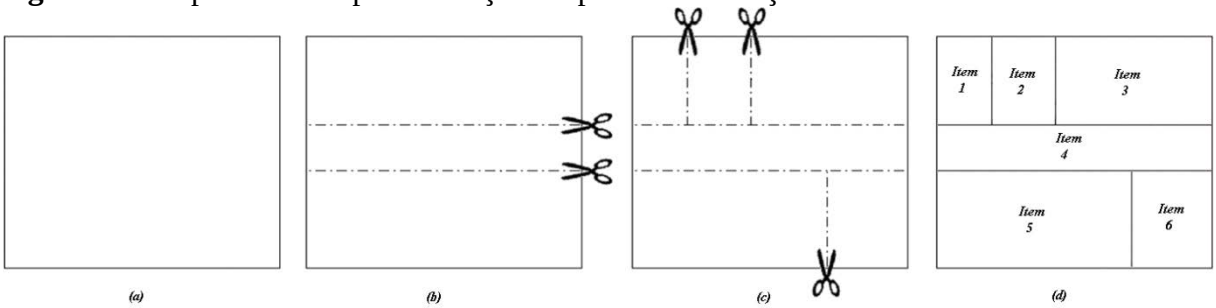
Figura 11 - Padrão não guilhotina



Fonte: Elaboração do próprio autor

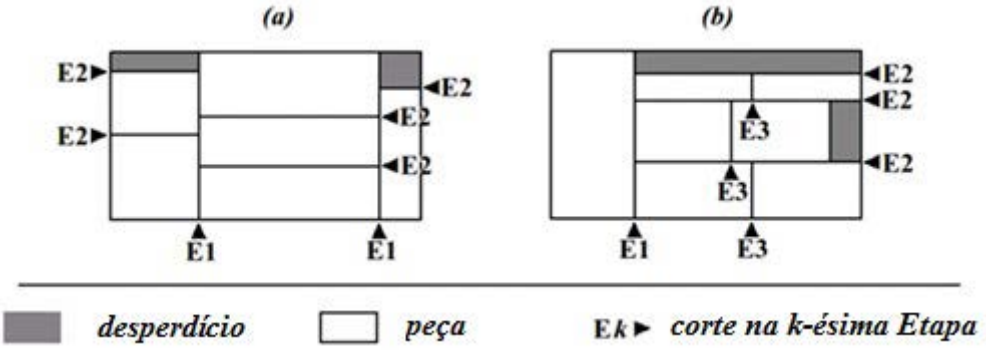
Alguns processos de corte industrial também limitam a maneira de produzir padrões de corte guilhotina. Na primeira etapa os cortes são realizados paralelos a um dos lados da mochila (ou chapa), depois, na seguinte etapa, ortogonal aos cortes prévios e assim sucessivamente. Isto é denominado como corte em etapas (ver Figura 12). Se existe um limite máximo imposto ao número de etapas de corte, denotado como *k*, o padrão guilhotina é chamado de: padrão de *k-etapas* (ver Figura 13), caso contrário se diz que é “sem etapas” (o padrão guilhotina sem etapas, é equivalente a, um padrão guilhotina de *k-etapas* definindo *k* o suficientemente grande, ver Figura 14).

Figura 12. Etapas de corte para alcançar um padrão de solução.



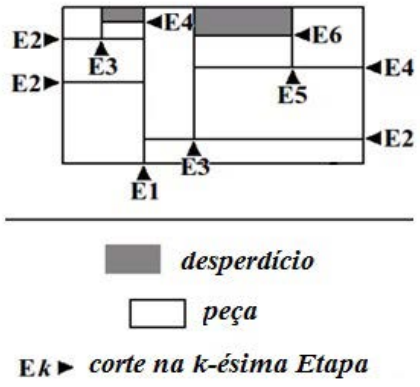
Fonte: Elaboração do próprio autor
(a) Mochila original,
(b) Primeira etapa de corte em direção horizontal,
(c) Segunda etapa de corte em direção vertical,
(d) Peças obtidas depois de duas etapas de corte.

Figura 13 - Padrões de corte tipo guilhotina de duas e três etapas



Fonte: Elaboração do próprio autor
(a) Padrão de corte tipo guilhotina de 2-etapas.
(b) Padrão de corte tipo guilhotina de 3-etapas.

Figura 14 - Padrão de corte tipo guilhotina sem etapas (ou de *k*-etapas, sendo *k* um número relativamente grande).

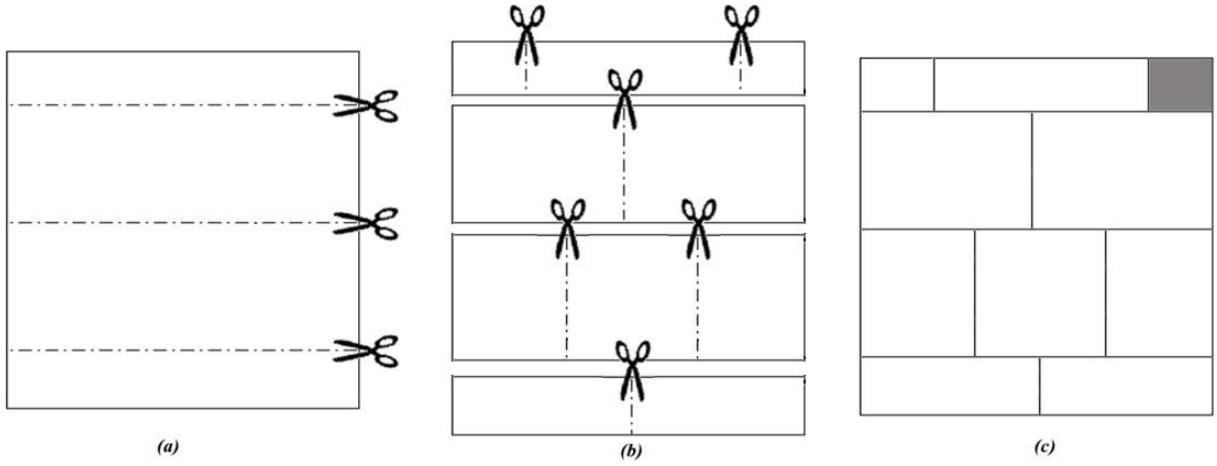


Fonte: Elaboração do próprio autor

Usualmente na indústria o maior número de aplicações encontradas é quando o número de etapas de corte é igual a dois (*k = 2*, ver Figura 15). Além disto, no corte em etapas é definido

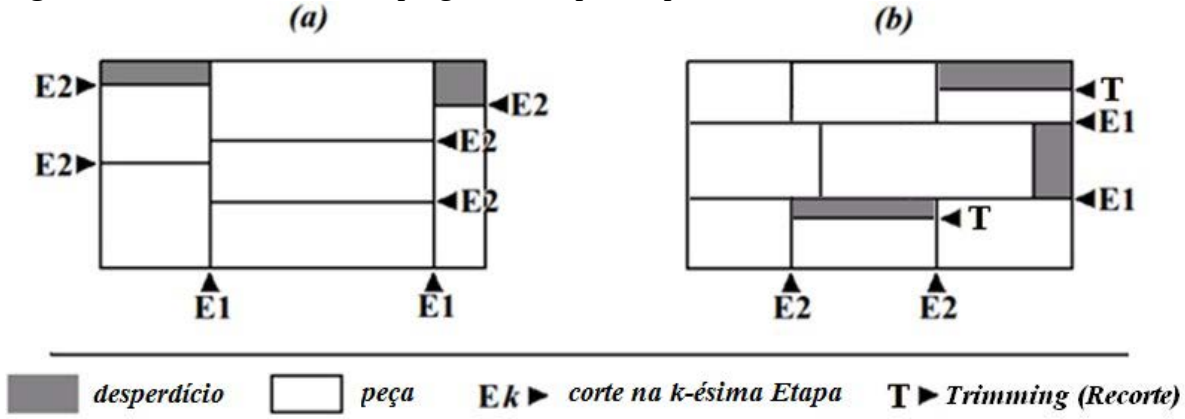
o uso do “recorte”, o qual é realizado ao final das etapas de corte, as peças nas suas dimensões demandadas não têm sido alcançadas, e é necessário um processo de recorte para obter a peça. Portanto, ao corte em etapas pode incluir ou não o processo de recorte (ver Figura 16).

Figura 15 - Padrão de corte tipo guilhotina de duas etapas



Fonte: Elaboração do próprio autor
(a) Primeira etapa de corte em direção horizontal.
(b) Segunda etapa de corte em direção vertical.
(c) Peças obtidas depois de duas etapas de corte.

Figura 16 - Padrões de corte tipo guilhotina por etapas sem e com recorte

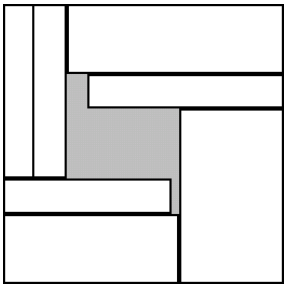


Fonte: Elaboração do próprio autor
(a) Padrão de corte tipo guilhotina de 2-etapas sem recorte.
(b) Padrão de corte tipo guilhotina de 2-etapas com recorte.

Por outro lado, alguns processos industriais não estão condicionados ao uso de guilhotinas para o corte bidimensional, tendo assim uma tecnologia para realizar os padrões de corte com dois graus de liberdade (duas dimensões geométricas). Obtendo assim, cortes tipo não guilhotina e por sua vez os padrões de corte não guilhotina, nos quais podemos encontrar dois tipos. O primeiro, denominado *padrão de primeira ordem*, consiste num padrão de corte

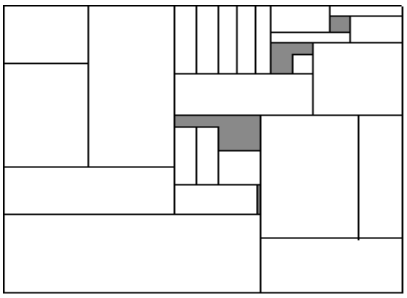
gerado a partir do “corte estampado” (quatro cortes ortogonais que produzem cinco novos retângulos) ilustrado na Figura 11. O mesmo padrão junto com os cortes tipo guilhotina em qualquer dos retângulos gerados continua sendo um *padrão de primeira ordem* (ver Figura 17). Igualmente, um corte estampado deste tipo aplicado a qualquer dos retângulos gerados (aplicação recursiva do mesmo) segue gerando um padrão de primeira ordem como se ilustra na Figura 18.

Figura 17 - Padrão de corte não guilhotina de primeira ordem, resultante do corte estampado e cortes guilhotina.



Fonte: Elaboração do próprio autor

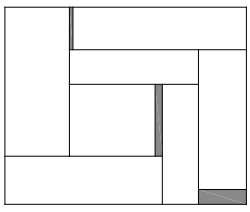
Figura 18 - Padrão de corte não guilhotina de primeira ordem, resultante dos cortes estampados aninhados e cortes guilhotina.



Fonte: Elaboração do próprio autor

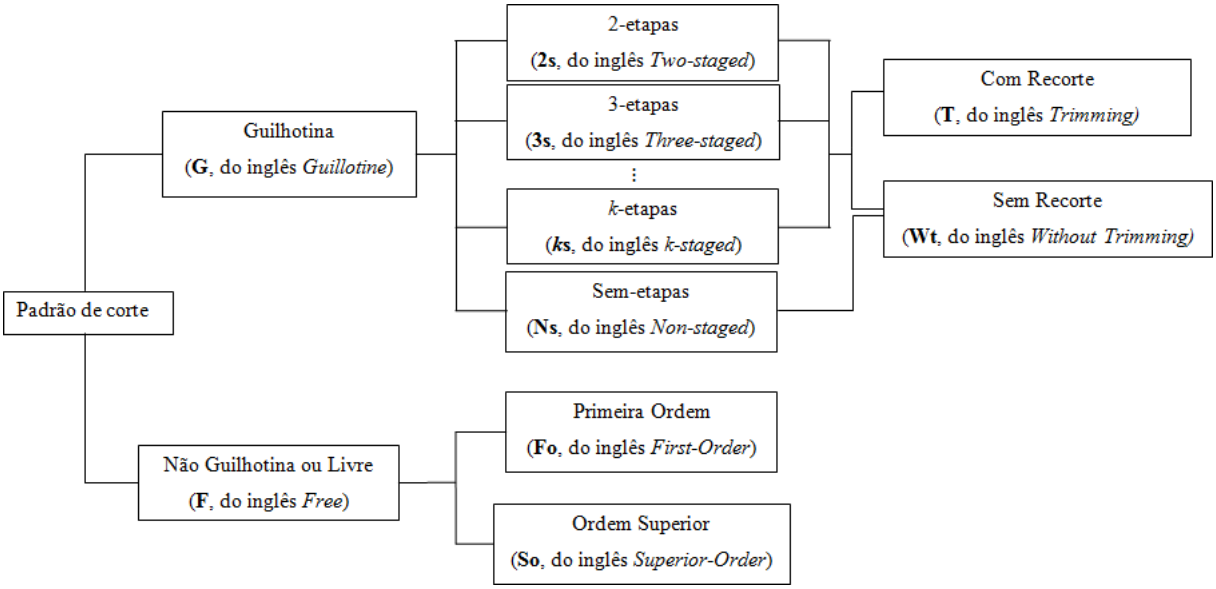
O segundo tipo de padrão, denominado *padrão de ordem superior*, é um padrão de corte não guilhotina que não se consegue alcançar usando unicamente cortes estampado e cortes tipo guilhotina, ou seja, é um padrão de corte livre, como se ilustra na Figura 19.

Figura 19 - Padrão de corte de Ordem Superior



Fonte: Elaboração do próprio autor

Figura 20 - Padrões de corte bidimensionais

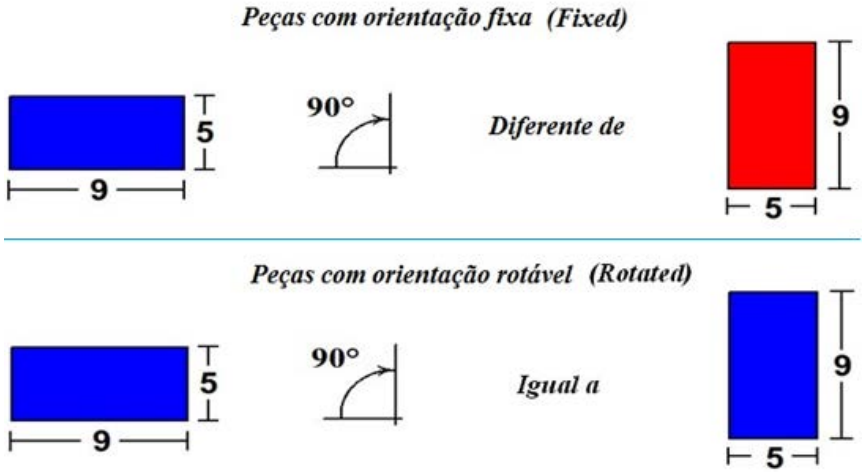


Fonte: Elaboração do próprio autor

- Características das peças bidimensionais

Lodi et al. (1999) classificam as características das peças em: sua orientação, seu valor (ou beneficio) e sua demanda (ou número de exemplares requeridos). Em diferentes cenários da indústria estas características podem estar condicionadas, além disto, várias características às vezes podem estar restritas, ou seja, cada combinação destas gera uma variante do problema.

Figura 21 - Peças com orientação fixa ou variável

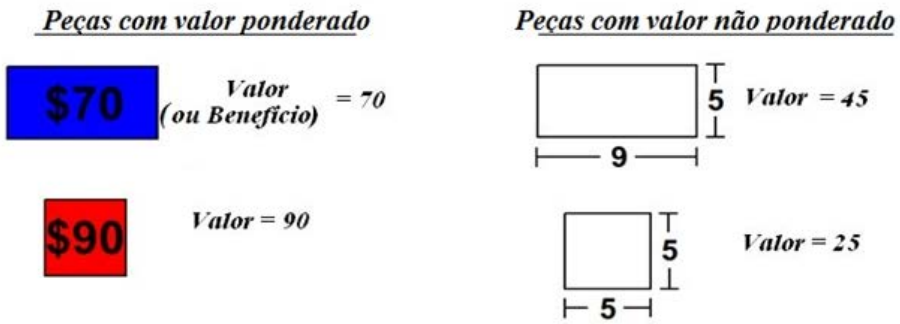


Fonte: Elaboração do próprio autor

Existem restrições inerentes à orientação das peças (a possibilidade de que as peças possam girar 90° ou não, ver Figura 21). Encontrando assim dois cenários: as peças podem girar 90° (**R**, do inglês *Rotated*) ou sua orientação é fixa (**Fx**, do inglés *Fixed*).

Os valores das peças (beneficio que oferece empacotar uma determinada peça) podem estar relacionados diretamente com sua área ou não (ver Figura 22). Existindo dois cenários que incidem diretamente na função objetivo do problema: os itens têm valores de beneficio ponderados, diferentes a sua área (**Wg**, conhecido na literatura como *Weighted output maximization*) ou os itens que têm valores de beneficio iguais a sua área (**Ug**, conhecido na literatura como *Unweighted output maximization*).

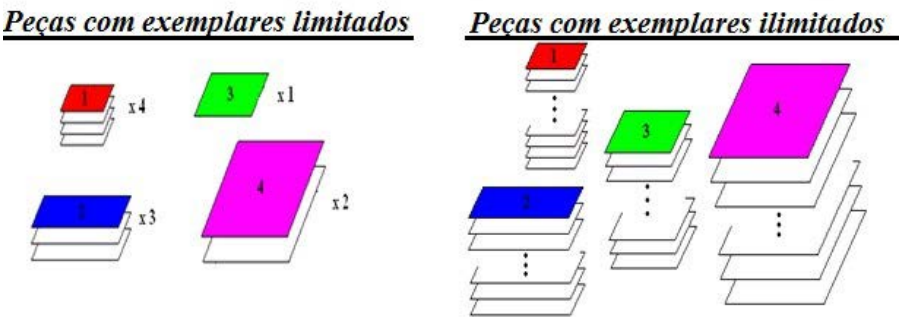
Figura 22 - Peças com valor ponderado ou não ponderado



Fonte: Elaboração do próprio autor

A demanda das peças (ou número de exemplares existentes do mesmo tipo de peça) pode estar ou não limitada (ver Figura 23). Quando não existe um limite máximo de copias por tipo de peça, diz-se que o problema é do tipo irrestrito (**Uc**, do inglês *Unconstrained problem*) e quando há uma limitação do número de exemplares por tipo de peça, diz-se que o problema é do tipo restrito (**C**, do inglês *Constrained problem*).

Figura 23 - Peças com exemplares limitados e ilimitados



Fonte: Elaboração do próprio autor

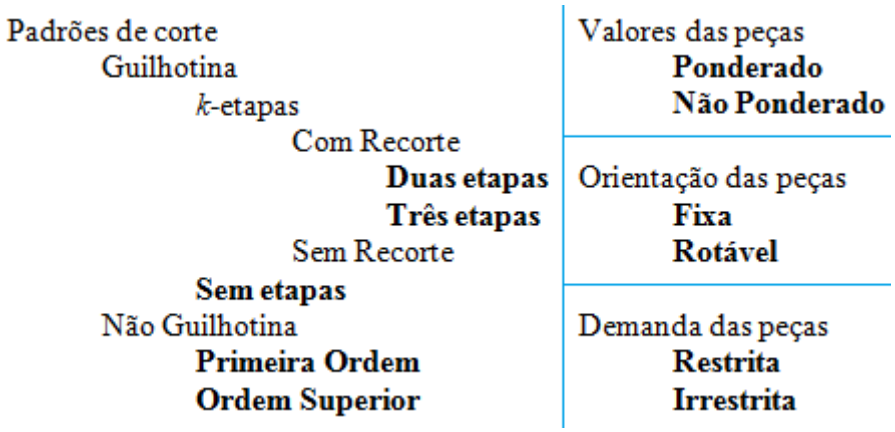
- Delimitação do problema da mochila tratado neste estudo

Anteriormente foi utilizada a classificação do Wäscher et al. (2007) para dividir este estudo em três tipos distintos de problema, em especial para o problema da mochila bidimensional

que é o primeiro a ser tratado neste documento; foi caracterizado usando as classificações do Morabito e Arenales (1996) para descrever os padrões de corte e do Lodi et al. (1999) para detalhar as características das peças. A combinação destas três tipologias, gera uma grande variedade de tipos dos problemas. Um tipo de problemas poderia ser aquele que estão condicionados aos padrões de corte guilhotina de duas etapas sem recorte, as peças têm uma orientação fixa, o valor das peças é ponderado (*weighted*) e a demanda das peças é restrita. Utilizando uma notação similar à usada nas classificações, escreve-se que o tipo de problema enunciado é: 2D-2s|C|Wg|Fx|Wt|SKP, do inglês *Two-dimensional Two-Staged Constrained Weighted Fixed Single Knapsack Problem without trimming*. Este tipo de problema em especial é um dos quatro que fazem parte do trabalho apresentado por Silva et al. (2010). Como vemos o número total de combinações é grande, sendo cada combinação um tipo de problema. Os estudos normalmente tentam resolver entre quatro ou oito tipos diferentes, porém também há estudos que unicamente tentam resolver um tipo de problema.

Neste trabalho serão estudados quarenta (ver Figura 24) tipos de problemas da mochila bidimensional, produto da combinação entre os padrões de corte tipo guilhotina de duas e três etapas considerando o processo de recorte, os padrões de corte tipo guilhotina sem etapas, os padrões de corte tipo não guilhotina de primeira ordem e ordem superior, peças com orientações fixas e variáveis, peças com valores ponderados e não ponderados, e peças com demanda irrestrita e restrita.

Figura 24 - Características do problema da mochila levadas em conta neste trabalho



Fonte: Elaboração do próprio autor

Apesar, de que parece ambicioso tentar resolver um número tão grande de tipos de problemas, depois veremos que uma metodologia de solução adequada pode resolver vários destes, ao aplicar somente pequenos ajustes que façam cumprir as condições do problema em questão.

Por outro lado, é importante mencionar que existe uma grande quantidade de trabalhos sobre estes problemas e a maioria baseados em aplicações reais da indústria, já que se está considerando um número suficiente de características para estudar e resolver o problema prático.

2.2.2 Restrições adicionais ao problema da embalagem

No problema da embalagem os critérios que são considerados para definir as restrições adicionais são os padrões de corte e empacotamento e as características das peças. As definições dos padrões de corte e empacotamento são as mesmas utilizadas para o problema da mochila. Deve-se notar que no problema de *Bin Packing* são usadas outras características das peças, em especial, as restrições de posicionamento entre peças, já que podem existir condições onde uma peça não deve ser empacotada ao lado de outra peça dentro do mesmo objeto. Isto acontece ao resolver o problema de empacotamento em paletes (*pallets*), que é comumente reduzido a um problema de *Bin-packing* bidimensional (o problema da embalagem) e se adicionam as restrições de posicionamento entre peças, devido a que, se por exemplo estamos empacotando produtos de abastecimento, não é permitido colocar produtos de limpeza ao lado (no mesmo *pallet*) de produtos alimentícios. As restrições de posicionamento entre peças são inerentes aos problemas de empacotamento em *pallets* ou ao problema de *Bin-packing* tridimensional, como veremos depois ambos não fazem parte deste estudo.

Como os padrões de corte e empacotamento são os mesmos definidos anteriormente, resta enunciar quais características das peças são levadas em conta. Neste estudo unicamente se consideram as restrições de orientação das peças, ou seja, se as peças têm uma orientação fixa ou se é permitido a rotação de 90° das peças. Por outro lado, é importante ressaltar que no problema da embalagem define-se uma classe de problemas onde as características dos objetos (chapas) são: pode existir um único tipo de objeto (todas as chapas têm as mesmas medidas, comprimento e largura), o fornecimento de objetos se assume ilimitado (ou pelo menos suficientemente grande para acomodar todas as peças), pode existir um único tipo de objeto e não há restrições de posicionamento das peças dentro das chapas (num objeto pode colocar qualquer tipo de peça) e pode existir um único tipo de objeto, onde todos os objetos têm o mesmo valor. Portanto, o objetivo do problema é minimizar o número de chapas necessárias para empacotar todas as peças demandadas.

- Delimitação do problema da embalagem tratado neste estudo

Neste trabalho se estudarão unicamente dois tipos de problemas resultantes da combinação entre os padrões de corte guilhotina sem etapas e as peças com orientações fixas e variáveis. Existem poucos trabalhos na literatura sobre o problema da embalagem com restrições de corte guilhotina, isto não se deve a pouca aplicabilidade deste tipo de problema, se não porque, o problema da embalagem tem sido mais estudado para resolver o problema de empacotamento em *pallets* o qual por definição utiliza padrões de corte tipo não guilhotina. Por esta razão, o estudo do problema com padrões guilhotina foi renegado para segundo plano. Além disto, neste trabalho não se estende ao estudo do problema incluindo restrições de padrão de corte tipo guilhotina em etapas (2-etapas, 3-etapas, etc.), porque para este tipo de problemas não se conhecem cenários reais.

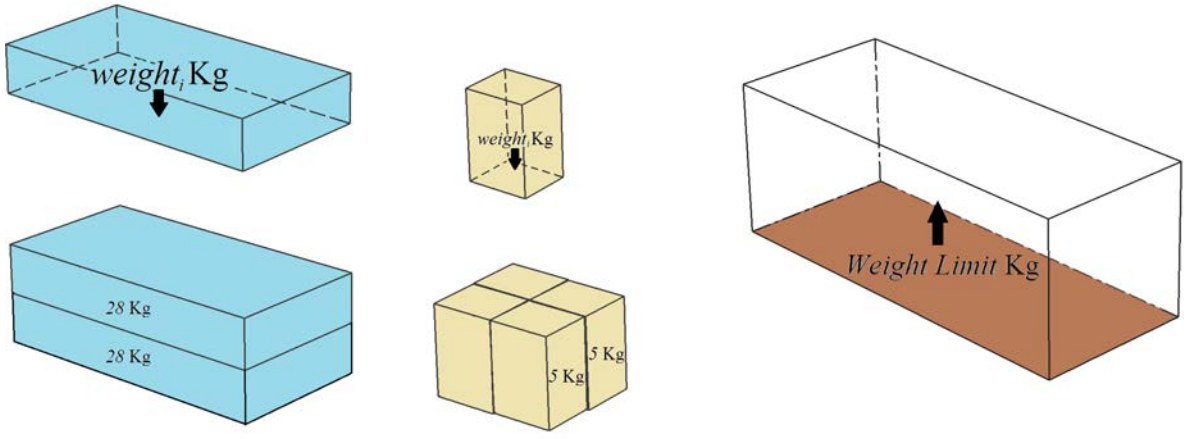
2.2.3 Restrições adicionais para o problema de carregamento do contêiner

Para os problemas de carga dos contêineres aparece a classificação apresentada por Bortfeldt e Wäscher (2013), a qual se divide nos critérios de restrições relacionadas com o contêiner, restrições relacionadas com as caixas, restrições relacionadas com o carregamento, restrições de posicionamento e restrições relacionadas com o padrão de empacotamento.

- Restrições relacionadas com o contêiner

Existem duas restrições relacionadas com o contêiner: o limite de peso e a distribuição do peso. A restrição do limite de peso consiste que o peso total das caixas empacotadas não ultrapasse o máximo permitido pelo contêiner (ver Figura 25).

Figura 25 - Limite máximo de peso suportado pelo contêiner



Fonte: Elaboração do próprio autor

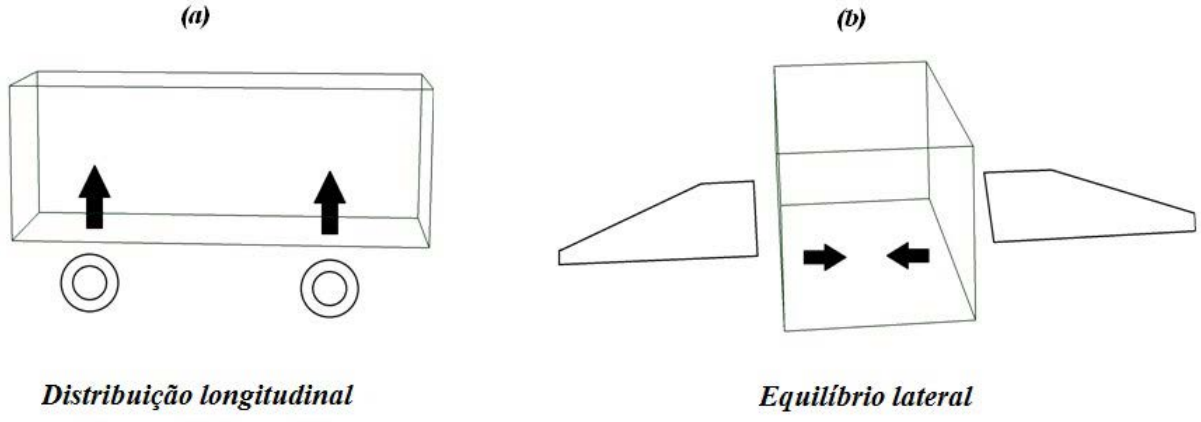
As restrições de distribuição do peso (ou restrições de balanço da carga) exigem que o peso da carga seja distribuído ao longo do piso do contêiner, esperando que um empacotamento com peso balanceado reduza as possibilidades do deslocamento da carga enquanto o contêiner se encontra em movimento (ver Figura 26).

- Restrições relacionadas com as caixas

Existem três restrições relacionadas com as caixas: restrições de caixas com prioridades de empacotamento, restrições inerentes às orientações das caixas e restrições de empilhamento das caixas.

As restrições de caixas com prioridades de empacotamento se derivam das condições práticas, onde, empacotar uma caixa pode ser mais desejável que a outra, por motivos (ou prioridades) como: prazos de entrega, requisitos relacionados com a frescura ou a vida útil dos produtos. Para isto, se definem duas classes de prioridades: prioridades absolutas e prioridades relativas. As prioridades absolutas consistem que uma caixa com este tipo de prioridade sempre deve ser empacotada, enquanto que as caixas com prioridade relativa têm um valor (ponderado ou não ponderado) do atrativo ou benefício que resulta empacotar este produto.

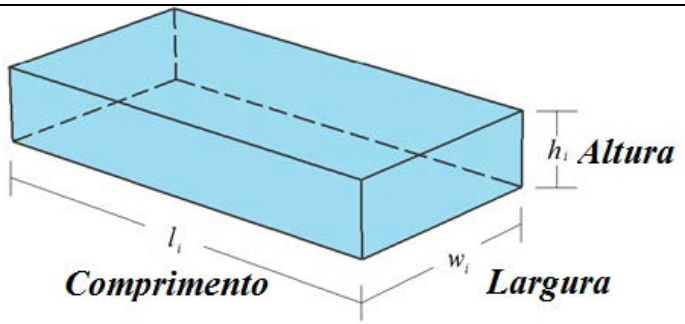
Figura 26 - Distribuição do peso dentro do contêiner

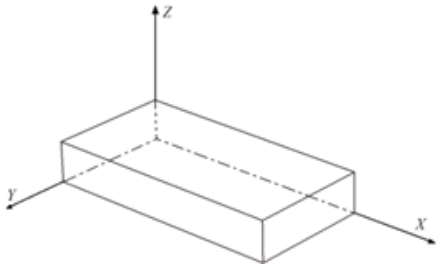
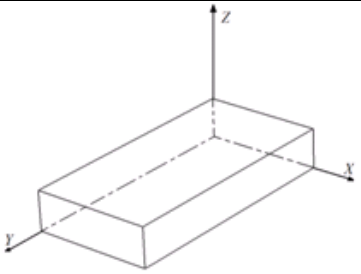
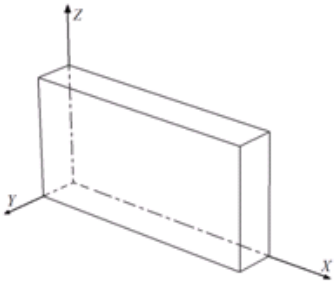
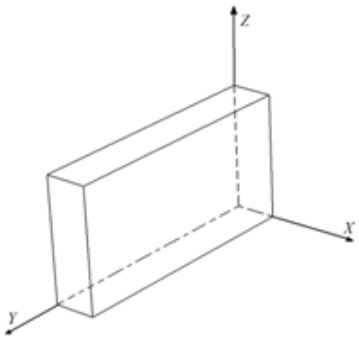
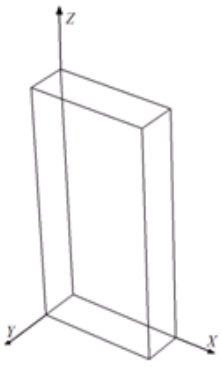
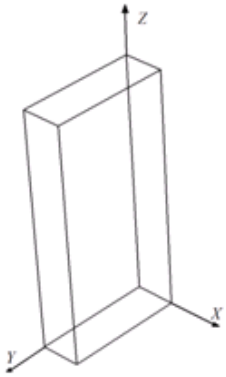


Fonte: Elaboração do próprio autor
(a) Distribuição do peso longitudinalmente (quando o contêiner representa o compartimento de um caminhão),
(b) Distribuição do peso lateralmente (em especial quando o contêiner representa o compartimento de um avião).

As restrições inerentes às orientações das caixas são baseadas, a princípio, que cada dimensão de uma caixa pode servir como altura, dando lugar a três orientações verticais.

Figura 27 - Possíveis orientações verticais e horizontais da peça



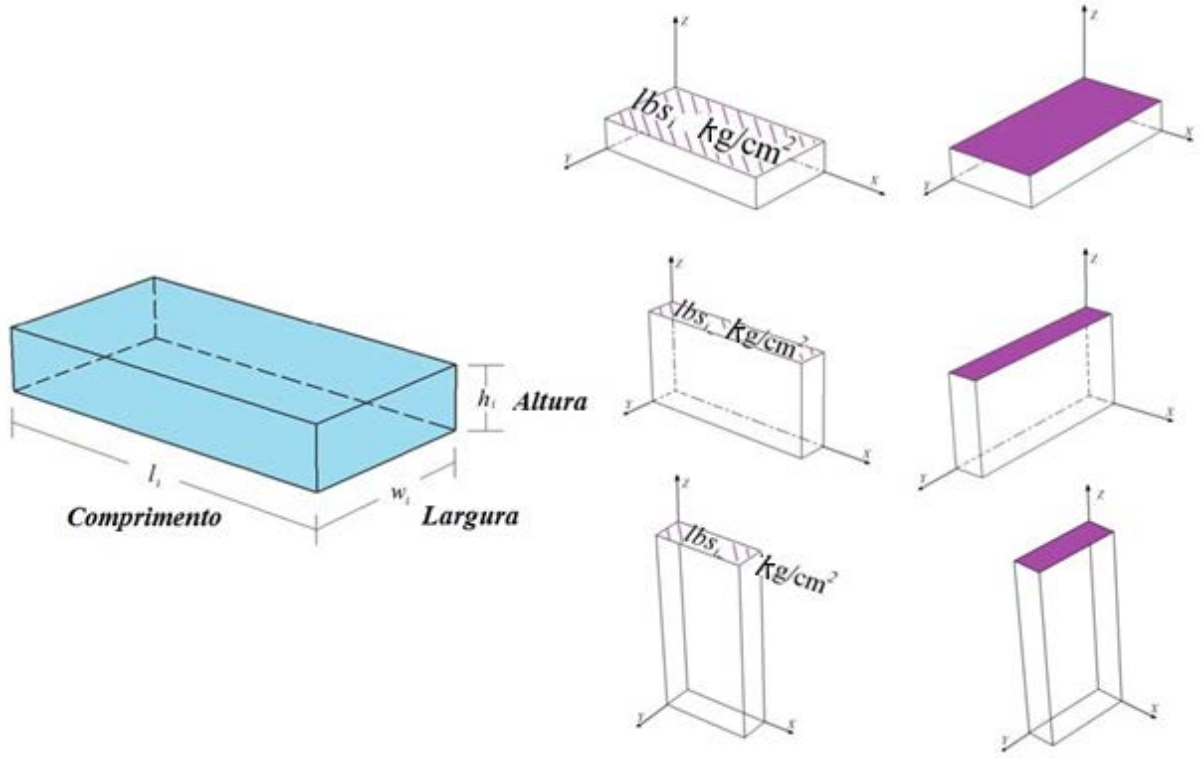
Orientação Vertical	**Orientação Horizontal**	
Altura	Comprimento	Largura
		
Largura	Comprimento	Altura
		
Comprimento	Largura	Altura
		

Fonte: Elaboração do próprio autor

Fixa-se uma dimensão da caixa em particular, como a altura, a orientação vertical da caixa estará definida. Portanto, unicamente restam duas possíveis dimensões para alinhar horizontalmente às paredes do contêiner (orientação horizontal da caixa), obtendo-se assim um total de seis orientações, segundo as quais uma caixa pode ser empacotada ortogonalmente num contêiner (ver Figura 27). Porém, na prática, o número de orientações permitidas de uma caixa pode ser restringido tanto em direção vertical como horizontal. Em geral, existem dois casos mais comuns: As caixas têm uma orientação fixa tanto vertical como horizontal (é dizer, as caixas não podem girar) e que não exista nenhuma restrição em geral a respeito da orientação vertical e horizontal das caixas (BISCHOFF; RATCLIFF, 1995).

As restrições de empilhamento das caixas limitam como as caixas podem ser localizadas uma sobre outra, uma aproximação geral a esta restrição é limitar o peso máximo que se pode aplicar por unidade de superfície à face suporte da caixa (ver Figura 28), isto quer dizer, que cada face da caixa tem um limite de resistência ao empilhamento (usualmente medido como peso por unidade de área, kg/cm^2).

Figura 28 - Limite de resistência ao empilhamento (*lbs*, do inglês *load-bearing strength*)



Fonte: Elaboração do próprio autor

Porém, esta aproximação assume que a pressão é aplicada uniformemente sobre a superfície (face) superior da caixa de suporte e não reflete o fato de que - devido a sua construção - a

caixa poderia ser capaz de suportar uma pressão maior nas bordas que no centro de sua face superior. Além disto, a rigidez da face superior de uma caixa, determina como realmente é transmitido o peso de uma caixa que se coloca na parte superior desta. Se a face superior da caixa localizada abaixo é feita de material macio (como papelão), então o peso será transmitido - mais ou menos – diretamente para baixo unicamente na superfície (área) que esteja em contato. Se a face superior é composta de um material muito rígido (como uma chapa de metal) então, o peso se distribuirá sobre toda a face superior da caixa de suporte (BISCHOFF, 2006).

- Restrições relacionadas com o carregamento

Existem duas restrições relacionadas com o carregamento: restrições de envio completo e restrições de localização.

As restrições de envio completo se derivam das condições práticas, onde, as caixas podem estar agrupadas em subconjuntos por motivos de funcionalidade ou gerenciamento. Isto significa que empacotar uma caixa de um subconjunto implica ter que empacotar a totalidade das caixas pertencentes a esse subconjunto e de maneira inversa, não empacotar uma caixa de um subconjunto exige não poder empacotar nenhuma caixa desse subconjunto, isto geralmente acontece em situações como o empacotamento de peças de mobiliário (unidades de cozinha, armários, etc.).

As restrições de localização surgem unicamente em problemas com múltiplos contêineres. Exigindo que os elementos de um subconjunto particular de caixas, devem ir no mesmo contêiner, por exemplo, quando um subconjunto vai ser enviado ao mesmo destino ou quando um cliente quer receber as peças pedidas num único envio e não como um envio de peças.

- Restrições de posicionamento

As restrições de posicionamento condicionam que a localização das peças dentro do contêiner seja em termos absolutos (ou seja, onde as peças devem ou não ser empacotadas dentro do contêiner) ou em termos relativos (ou seja, donde as peças devem ou não ser localizadas com relação a outras peças).

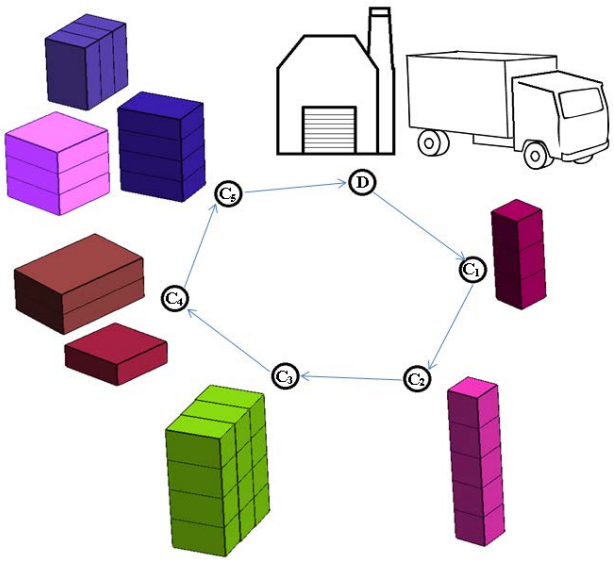
As restrições em termos absolutos demandam que determinadas peças sejam localizadas (ou não) numa posição em particular ou numa área em particular do contêiner. Estas restrições são normalmente impostas pelo tamanho, peso ou pelo conteúdo da caixa. Por exemplo, caixas

muito pesadas serão possíveis de descarregar unicamente se localizarem perto da porta do contêiner ou ao contrário, por limitações de manuseio uma caixa de grandes extensões e peso que impedirá o fácil acesso, será desejável localizá-la na parte posterior do contêiner.

As restrições em termos relativos podem por um lado, demandar que um grupo de caixas seja localizado relativamente junto dentro do contêiner ou ao menos, a uma determinada distância uma da outra, por exemplo, quando esse grupo de caixas representa um cliente, é desejável para uma fácil verificação e descarregamento deste. Por outro lado, estas restrições também podem demandar que certas caixas (ou grupo de caixas) não sejam localizadas juntas ou na proximidade destas, dado que podem afetar-se umas às outras de maneira negativa, por exemplo, posicionar alimentos nas proximidades de produtos combustíveis.

Dentro das restrições de posicionamento aparecem os cenários *multi-drop*, que resultam ser uma combinação de restrições de posicionamento em termos absolutos e relativos. Uma situação *multi-drop* se caracteriza pelo fato de que subconjuntos de caixas devem ir a diferentes clientes e o roteiro de visita destes já está determinado. As caixas de um subconjunto não somente deverão localizar-se na proximidade das outras, além disto o arranjo de todos os subconjuntos dentro do contêiner deverá refletir a sequência de acordo com o roteiro de entrega, de forma tal que, ao alcançar cada destino não sejam necessárias operações ociosas de descarregamento e recarregamento.

Figura 29 - Situação *multi-drop*. Subconjuntos de caixas devem ir a diferentes clientes e o roteiro de visita destes já está determinado

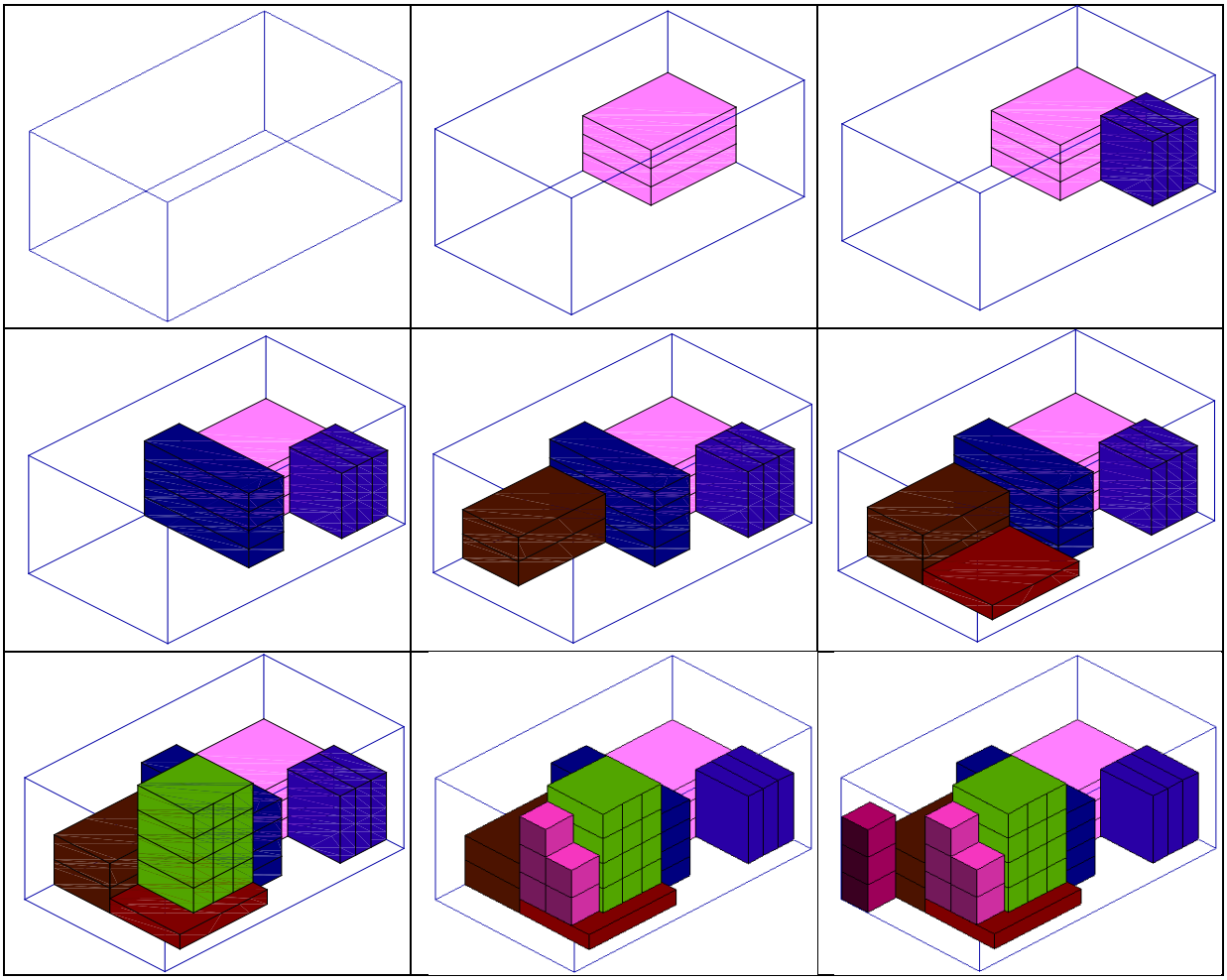


Fonte: Elaboração do próprio autor

A Figura 29 ilustra uma situação *multi-drop* através de um grafo dirigido, onde os nós representam os clientes (os nós C_i) e o ponto de depósito (o nó D), as arestas representam a ordem na qual devem ser visitados os clientes, iniciando a roteiro do depósito. Neste as caixas são agrupadas por clientes (ou caixas com o mesmo destino de desembarque).

A Figura 30 ilustra o processo de empacotamento da situação *multi-drop* proposta na Figura 31, a embalagens se realizam no sentido contrário do roteiro de visita dos clientes e ao mesmo tempo se deve verificar e garantir que não sejam necessárias operações ociosas de descarregamento e recarregamento de caixas.

Figura 30. Exemplo de empacotamento *multi-drop*. Empacotamento da situação proposta na Figura 29.



Fonte: Elaboração do próprio autor

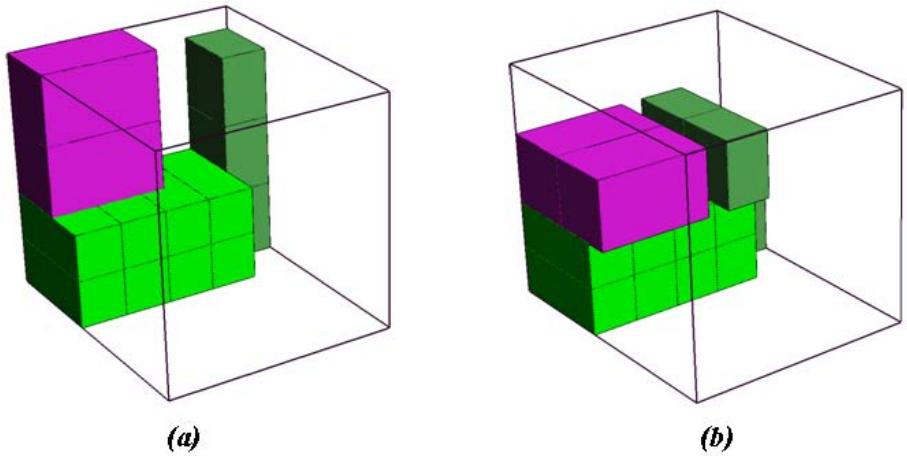
- Restrições relacionadas com o padrão de empacotamento

Existem duas restrições relacionadas com o padrão de empacotamento: restrições de estabilidade e restrições de complexidade. Ambas as restrições, referem-se às propriedades desejáveis ou necessárias do arranjo final das caixas dentro do contêiner.

As restrições de estabilidade do padrão de carga frequentemente são consideradas as mais importantes que o mesmo objetivo geral de maximizar o espaço utilizado, pois um empacotamento instável pode resultar em danos à carga e inclusive pode chegar a ferir o pessoal encarregado das operações de carga e descarga das caixas. Em situações práticas, a estabilidade do padrão de carga pode ser alcançada ao adicionar suportes (ou abraçadeiras) ou preenchendo os espaços vazios restantes com um material como a espuma.

Com relação à estabilidade do padrão de carga, podem-se distinguir dois tipos: estabilidade vertical e estabilidade horizontal. A estabilidade vertical, também chamada: estabilidade estática (DE CASTRO SILVA et al., 2003) evita que as caixas caiam no solo do contêiner ou sobre a superfície de outras caixas. Esta situação aparece quando o contêiner não se encontra em movimento e descreve a capacidade do padrão de carga de resistir à força da gravidade (JUNQUEIRA et al., 2012).

Figura 31 - Estabilidade vertical (ou estática) do padrão de carga



Fonte: Elaboração do próprio autor
(a) Padrão de carga com estabilidade vertical exigindo suporte completo da base de todas as caixas.
(b) Padrão de carga com estabilidade vertical exigindo uma fração mínima de suporte da base das caixas, isto permite que algumas caixas que sobressaiam (ou seja, parte de sua base fica pendurando ou sem suporte).

Os problemas com estabilidade vertical acostumam demandar que a base de cada caixa deve ser suportada (na totalidade ou parcialmente), seja pelo solo do contêiner ou espaçamento uniforme (é dizer, um espaço com o mesmo nível de altura) proporcionado pelas superfícies

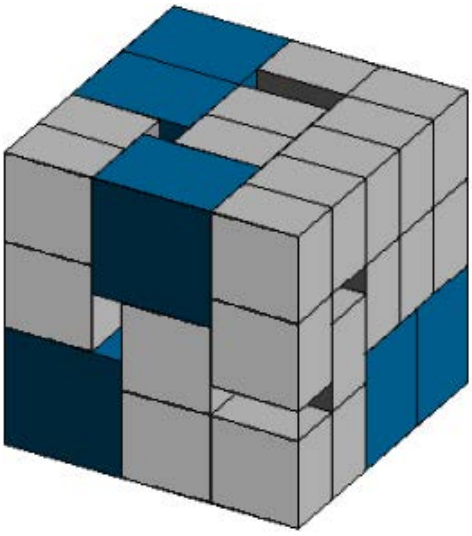
superiores de outras caixas. O suporte necessário pode exigir que a superfície da base da caixa devesse estar completamente em contato ou especificar uma fração mínima de contato (ver Figura 31).

Quando se demanda estabilidade de suporte completo (ou seja, o 100% da base da caixa deve estar suportada) pode-se garantir a estabilidade vertical do padrão de carga. Ao contrário, quando se relaxa a porcentagem da base da caixa, isto pode derivar a alcançar padrões de carga instáveis, e necessitaria do uso de abraçadeiras ou preencher os espaços vazios resultantes com materiais como espuma, para poder garantir a estabilidade vertical.

A estabilidade horizontal (ou também chamada de estabilidade dinâmica) garante que as caixas não se mexam significativamente enquanto o contêiner está em movimento. Por tanto, esta se refere à capacidade do padrão de carga de resistir à inércia das caixas (JUNQUEIRA et al., 2012). A estabilidade horizontal completa é alcançada quando se garante que cada peça empacotada está adjacente (horizontalmente ao lado) de outra caixa ou de uma parede do contêiner (BISCHOFF; RATCLIFF, 1995).

As restrições de complexidade do padrão de empacotamento, garantem que a embalagem possa ser alcançada através da tecnologia disponível. Por um lado, um padrão de carga pode não ser aceitável para o carregamento manual de contêineres, porque esses padrões nem sempre podem ser visualizados de maneira que sejam entendidas corretamente pelo pessoal encarregado de fazer o carregamento (ver Figura 32) e sua embalagem pode demorar muito tempo (ver Figura 33).

Figura 32. Padrão de carga livre.



Fonte: Elaboração do próprio autor

Tecnologias de embalagem mecânicas e automáticas mais avançadas, por outro lado, nem sempre são adequadas para os padrões de carga complexos e podem requerer a participação de mão de obra adicional representando mais custos (ver Figura 34). As restrições de complexidade refletem as limitações dos recursos tecnológicos e humanos (BISCHOFF; RATCLIFF, 1995).

Figura 33. Carregamento/Descarregamento manual de um contêiner.

Fonte: Vaculex, Disponível em: <http://www.vaculex.com/Solutions/DistributionCenters.aspx>.

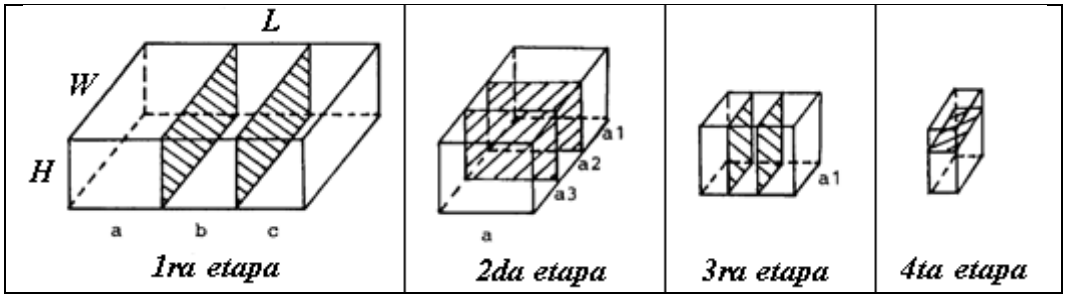
Figura 34. Carregamento/Descarregamento mediante tecnologias mecânicas o automáticas.

Fonte: Vaculex, Disponível em: <http://www.vaculex.com/Solutions/DistributionCenters.aspx>.

A restrição de complexidade considerada mais frequentemente é a restrição de padrões de corte guilhotina. Um padrão tipo guilhotina (mais exatamente: desenho guilhotinado, padrão de corte guilhotinado) representa um tipo de padrão de carga que pode ser facilmente descrito e empacotado. Diz-se que um padrão de carga é guilhotinado, se pode obter-se através de uma série de cortes paralelos às faces do contêiner (ver Figura 35). A geração dos padrões de

guilhotina não é amplamente abordada na literatura do problema de carregamento de contêineres, devido que estes nem sempre são adequados, uma vez que eles tendem a ser instáveis (WÄSCHER; BORTFELDT, 2013).

Figura 35. Padrão de carga guilhotinado, representação dos cortes paralelos às faces do contêiner (aplicados por etapas).



Fonte: Elaboração do próprio autor

- Delimitação do problema de carregamento do contêiner tratado neste estudo

Neste trabalho será estudado o problema de carregamento do contêiner considerando restrições que representam condições práticas do problema na vida real. Começando pela inclusão das restrições de orientação das peças que resulta ser a restrição mais levada em conta na literatura. Além disto, adicionar as restrições de limite de resistência ao empilhamento entre caixas, junto com a restrição de limite máximo de peso do carregamento suportado pelo contêiner. Mantendo a estabilidade da carga mediante as restrições de estabilidade vertical de suporte completo. Finalmente, adicionar as restrições de embalagens *multi-drop*, que são de grande interesse na indústria embora ainda tenha sido pouco estudado e as metodologias propostas parecem não explorar todas as características deste tipo de embalagem. Em geral, na literatura unicamente se encontra um trabalho que reúne todas estas restrições práticas (CESCHIA; SCHAERF, 2013).

3 PROBLEMA DA MOCHILA BIDIMENSIONAL

O problema de corte é considerado clássico dentro da pesquisa operacional devido a seu grande espectro de aplicação na indústria e sua alta complexidade tanto matemática como computacional. Neste trabalho é apresentado o problema da mochila bidimensional. Descreve-se o modelo matemático aplicado por diferentes grupos de pesquisa que estudam esta temática. Propõe-se um tipo de codificação para ser aplicada neste problema e se resolve mediante um algoritmo de otimização que combina as principais características de otimização por enxame de partículas, busca em vizinhança variável e algoritmos genéticos. Para comprovar a eficiência da metodologia proposta foram realizados testes usando instâncias testes da literatura especializada, a fim de analisar e compara com os métodos de resolução apresentados no estado da arte do problema.

3.1 INTRODUÇÃO

O problema da mochila bidimensional aparece quando o material utilizado é um objeto retangular, no qual se devem localizar peças retangulares menores, das quais se conhece o tamanho e o custo associado (ou beneficio); o objetivo é maximizar o valor das peças cortadas.

As características deste problema são as seguintes:

i) O custo associado às peças pode ou não estar relacionado com sua área; se o custo é igual à área da peça estase resolvendo o problema sem pesos (custos não ponderados do inglês *unweighted*) e se o custo é diferente da área do item se está resolvendo o problema com pesos (custos ponderados do inglês *weighted*).

ii) A orientação das peças a ser localizadas, ou seja, a orientação de uma peça é fixa se uma peça de altura h e largura w é diferente de uma peça de largura w e altura h (orientação fixa do inglês *fixed version*). Se considerarmos que as dimensões (h, w) e (w, h) representam as dimensões da mesma peça, estamos abordando o problema com rotação (orientação variável do inglês *rotated version*).

iii) Os padrões de corte podem ser do tipo guilhotina ou não guilhotina, no tipo guilhotina se diferenciam: com duas e três etapas de corte (*two-staged and three-staged versions*), e sem etapas de corte (*non-staged version*), adicionalmente em estes se

permite um pós-processo de recorte (*with trimming version*). Enquanto que nos tipos não guilhotina se diferenciam os padrões de primeira ordem (*first order version*) e de ordem superior (*superior order version*).

iv) A demanda de peças ou limite máximo do número de peças a cortar de cada tipo, pode ser ilimitada (*unconstrained version*) ou restrita (*constrained version*).

Para o problema da mochila bidimensional com padrões tipo guilhotina um resumo do estado da arte é o seguinte: (GILMORE; GOMORY, 1965; 1966) propõem um algoritmo recursivo exato sobre a base da programação dinâmica para resolver o problema. Este algoritmo é aplicável às versões onde as peças têm valores ponderados e não ponderados. De acordo com Herz (1972) foi proposto um método de busca recursiva de árvore, este método é mais eficaz que o algoritmo de Gilmore e Gomory (1965) para o problema onde as peças têm valores não ponderados, porém não se aplica aos casos com peças com valores ponderados.

Morabito et al. (1992) desenvolverem a heurística DH (do inglês *Depth-first search e Hill-climbing strategies*) baseada num processo de busca primeiro em profundidade e estratégias de aceitação de deterioração. O algoritmo KD (do inglês *Knapsack problem* usando *Dynamic programming*) foi apresentado por Fayard e Zissimopoulos (1995) para resolver o problema baseado na solução de problemas da mochila unidimensional, resultando eficiente para problemas de tamanho grande. Este algoritmo também usa uma estrutura de grafo, mas somente considera o primeiro nível da árvore, a diferença do DH que consiste em desenvolver uma árvore limitada por um parâmetro de profundidade (HERZ, 1972; CHRISTOFIDES; WHITLOCK, 1977; MORABITO et al., 1992).

Hifi, (2001) apresenta um algoritmo híbrido que combina o algoritmo DH e KD, isto permite desenvolver um algoritmo geral para o problema da mochila bidimensional irrestrita guilhotinada. O algoritmo proposto é testado para instâncias do problema de tamanho grande, tanto em problemas com pesos como sem pesos, obtendo excelente resultados.

Hifi e Zissimopoulos, (1996) propõe um algoritmo recursivo exato usando programação dinâmica baseada em eficientes limites inferiores e superiores para resolver o problema irrestrito. Young-Gun e Kang (2002) propõem uma melhora ao algoritmo Hifi e Zissimopoulos (1996) usando uma cota superior mais eficiente. Este é atualmente uno dos algoritmos exatos com melhor desempenho para resolver o problema irrestrito.

Existem duas técnicas gerais utilizadas para resolver os problemas restritos: *top-down* e *bottom-up*. Christofides e Whitlock (1977) propuserem originalmente o enfoque de *top-down*, o qual gera todos os possíveis padrões de solução em forma recursiva, dividindo cada chapa em duas novas. A maioria dos algoritmos exatos para resolver o problema irrestrito utiliza este enfoque. O enfoque de *bottom-up* gera todos os possíveis padrões de corte através da combinação de dois padrões horizontal ou verticalmente construídos, a partir de outros dois padrões. O enfoque *bottom-up* requer uma grande quantidade de memória, razão pela qual a implementação de um algoritmo deste tipo é pouco atrativa, embora Young-Gun et al. (2003) propuserem um algoritmo que parte de uma solução inicial de boa qualidade e utiliza o algoritmo construtivo *bottom-up* como estratégia para gerar os ramos, diminuindo o número de nós a explorar.

Por outro lado, um resumo do estado da arte do problema da mochila bidimensional com padrões de corte não guilhotina é o seguinte: Beasley, (1986) e Hadjiconstantinou e Christofides (1995) apresentam uma formulação de programação linear 0-1 para o problema, utilizando variáveis de decisão binárias para as posições na chapa onde serão cortadas as peças; desenvolverem relaxações Lagrangeanas e as utilizarem como limites nos procedimentos de busca na árvore correspondente (os limites são melhorados usando a otimização do subgradiente).

Em (TSAI et al., 1993) e (CHEN et al., 1995), o problema é formulado como um modelo linear binário utilizando as variáveis de decisão esquerda, direita, acima e abaixo, relativas à posição de cada par de peças que se cortarão da mochila (com restrições disjuntivas para as opções múltiplas); nestes se sugere resolver os modelos utilizando algoritmos *Branch-and-Bound* explorando estruturas particulares das mencionadas restrições. Outra formulação linear binária aparece em (BOSCHETTI et al., 2002; EGEBLAD; PISINGER, 2009), e uma formulação binária não linear foi apresentada em (BEASLEY, 2004).

Outra maneira exata de abordar o problema é utilizando *Branch-and-Bound* baseado no chamado enfoque de dois níveis (o primeiro nível seleciona o conjunto de peças a cortar sem levar em conta a camada de corte, o segundo nível revisa se existe uma camada de corte factível para as peças selecionadas) (BALDACCI; BOSCHETTI, 2007; CAPRARA; MONACI, 2004; FEKETE et al., 2007).

O método apresentado em (FEKETE et al., 2007) é baseado numa árvore de busca de dois níveis que combina o uso de uma estrutura de dado especial para caracterizar os cortes factíveis com limites superiores.

Em (BIRGIN et al., 2012) se utiliza o enfoque de fracionamento recursivo apresentado em (BIRGIN et al., 2010) para a carga de manufaturas em *pallets*, para os cortes bidimensionais ortogonais não guilhotina (ponderado e não ponderado) somente para gerar padrões de primeira ordem nos contêineres. Este enfoque recursivo de fracionamento combina versões refinadas de ambas as heurísticas recursivas de cinco blocos apresentadas em (MORABITO; MORALES, 1998; 1999) e a estratégia de corte em forma de ele (*L)* apresentadas em (LINS et al., 2003; BIRGIN et al., 2005) para cortar retângulos de retângulos maiores e peças em forma de ele (*L*).

Diferentes heurísticas baseadas em busca local aleatória, localização *bottom-left*, busca em grafos, etc., e diferentes meta-heurísticas baseadas em algoritmos genéticos, busca tabu, arrefecimento simulado, GRASP, etc., para gerar cortes bidimensionais não guilhotina para o problema restrito e irrestrito se encontram na literatura. Alguns exemplos recentes são (ALVAREZ-VALDÉS et al., 2007; GONÇALVES, 2007; HUANG; CHEN, 2007; CHEN; HUANG, 2007; BORTFELDT; WINTER, 2009; EGEBLAD; PISINGER, 2009; WEI et al., 2009).

Os métodos baseados em técnicas não lineares para empacotamento de retângulos dentro de regiões arbitrárias convexas, considerando diferentes tipos de restrições de posicionamento tem sido apresentadas em (BIRGIN et al., 2006a; BIRGIN et al., 2006b; BIRGIN; LOBATO 2010; MASCARENHAS; BIRGIN, 2010). A maioria destes estudos foi desenvolvida para o problema restrito, que pode ser mais interessante para certas aplicações práticas com relativa baixa demanda dos itens ordenados; embora, parte desses métodos poderiam não funcionar bem quando se resolve o problema irrestrito, especialmente aqueles com comportamento computacional altamente dependente do total de números de itens ordenados. Por outro lado, o problema irrestrito é particularmente interessante para aplicações do problema de *cutting-stock* com produção de grande escala e com itens fracamente heterogêneos (ou seja, relativamente poucos tipos de peças, porém muitas cópias do mesmo tipo), no qual o problema utiliza o procedimento de geração de colunas, como indicam muitos autores desde o estudo pioneiro de (GILMORE; GOMORY, 1965).

3.2 DESCRIÇÃO DO PROBLEMA

O problema da mochila bidimensional estudado neste trabalho se define como cortar de um retângulo que se denomina mochila (chapa, lâmina ou objeto) de altura *H* e largura *W*, um conjunto de *n* retângulos que se denominam peças (itens), cada peça *i* possui de altura h_i, largura w_i, número de exemplares b_i e custo associado c_i (aonde $i = 1,..., n$). A função objetivo é definida pela Equação 1 e consiste em maximizar o custo associado ao conjunto de peças cortadas, z_i é uma variável inteira que indica o número de exemplares da peça *i* que devem ser cortadas.

$max \quad \sum_{i=1}^{n} c_i \cdot z_i$	*(1)*

Sujeito a:

1 - As peças empacotadas não devem superar os limites da mochila.

2 - As peças não devem sobrepor-se entre elas.

3 - $z_i \leq b_i$, as peças atribuídas não devem superar a demanda por tipos de peças.

A Equação 1 e as Expressões 1 – 3 representam o problema geral.

Neste trabalho se estudam diferentes variantes que são encontradas na indústria. As características destas variantes podem ser incorporadas ao adicionar expressões ou modificar alguma destas.

As características deste problema são as seguintes:

i) Valores das peças: se o custo associado da peça não está relacionado com a área, a Expressão 1 representa perfeitamente esta variante. Por outro lado, se o custo associado da peça deve ser a área, deve ser agregada a Equação 2.

$c_i = h_i \cdot w_i$	*(2)*

ii) A orientação das peças: se a orientação das peças é fixa, não é necessário adicionar nenhuma expressão. Por outro lado, se é permitido rodar as peças 90 graus, deve-se realizar um pré-processo que identifique as peças *i* e *k* tais que $h_i = w_k$ e $w_i = h_k$, e para estas fazer $b_i = b_i + b_k$, (ou seja, agrupar por tipos de peça), além disto, se deve adicionar a Expressão 4.

4 - As peças podem rodar 90°

iii) Os padrões de corte: para os padrões tipo guilhotina se deve adicionar a expressão correspondente. A Expressão 5 para os padrões tipo guilhotina de duas etapas de corte, a Expressão 6 para os padrões tipo guilhotina de três etapas e a Expressão 7 para os padrões tipo guilhotina sem etapas. Por outro lado, para os padrões tipo não guilhotina, se deve adicionar a Expressão 8 se requeremos um padrão de corte não guilhotina de primeira ordem, caso contrário, se o padrão de corte é não guilhotina de ordem superior não é necessário adicionar nenhuma expressão.

5 - Só o uso de cortes guilhotina é permitido e duas etapas de corte no máximo.

6 - Unicamente o uso de cortes guilhotina é permitido e três etapas no máximo.

7 - Unicamente o uso de cortes tipo guilhotina é permitido.

8 - Unicamente o uso de corte tipo guilhotina e cortes tipo não guilhotina de primeira ordem.

iv) A demanda de peças ou o limite máximo de número de peças a cortar de cada tipo: se não existe um limite máximo de exemplares por tipo de peça se deve modificar a Expressão 3 por a Expressão 9. Por outro lado, foi imposto um limite máximo b_i de exemplares por tipo de peça não é necessária adicionar nenhuma expressão.

9 - $z_i \leq M_i$, onde M_i é um número o suficientemente grande. $M_i = \left\lfloor \frac{W \cdot H}{w_i \cdot h_i} \right\rfloor$

3.3 MODELO MATEMÁTICO

Distintos modelos matemáticos para representar os mesmos tipos de problema já têm sido propostos, com a finalidade de encontrar melhoras na solução através de uma modelagem eficiente, é por isto que neste estudo fazemos uma revisão dos diferentes modelos e se expõe somente dois porque são os mais próximos aos problemas propostos, além de apresentar resultados satisfatórios para o problema da mochila bidimensional.

Gilmore e Gomory, (1961; 1963) apresentam o primeiro modelo matemático para o problema de corte e empacotamento unidimensional. Gilmore e Gomory, (1961; 1963) estudam cuidadosamente o problema da mochila quando este é aplicado para o corte de peças em uma e duas dimensões.

Biro e Biros, (1985) caracterizam os padrões não guilhotina usando teoria de grafos. Beasley, (1986) estudou o problema da mochila com padrões de corte tipo não guilhotina, propondo um modelo de programação linear inteira para determinar as coordenadas discretas às quais os itens podem ser localizados. Um modelo similar foi introduzido por (HADJICONSTANTINOU; CHRISTOFIDES, 1995). Scheithauer e Terno (1993) propõem modelos para empacotamento de polígonos convexos e não convexos.

Fekete e Schepers, (1997) usam a teoria de grafos para determinar um empacotamento factível sem sobrepor as peças. Lodi et al. (2002; 2004) apresentam um modelo que manipula padrões formados por estantes (níveis), este modelo considera explicitamente restrições de corte tipo guilhotina (mas só uma parte dos padrões guilhotina são considerados). Beasley (2004) apresenta uma nova formulação de corte não guilhotina. Uma boa revisão dos modelos existentes está disponível em (LODI et al., 2004).

Um modelo de programação linear inteira mista (MILP, do inglês *Mixed Integer Linear Programming*) para o problema de empacotamento tridimensional em contêineres, com número polinomial de variáveis e restrições, é apresentado por (CHEN et al., 1995). Este modelo pode ser visto como uma extensão à técnica de modelagem proposta por (ONODERA et al., 1991). Para um problema de localização de blocos bidimensionais, o modelo tem como base a enumeração de todas as possíveis localizações relativas de cada par de peças. Os experimentos computacionais apresentados em (CHEN et al., 1995) mostram que o modelo proposto é muito ineficiente na solução de instâncias práticas de empacotamento. A mesma técnica foi usada por (DANIELS et al., 1994) para modelar o problema de empacotamento geral (bidimensional) de polígonos, neste mesmo caso, o uso direto do modelo prova ser ineficiente na prática. Benet al. (2008) apresentam um modelo baseado na caracterização e modelagem dos padrões guilhotina para o problema de empacotamento em tiras infinitas (*strip packing problem*), o qual pode ser facilmente adaptado para o problema da mochila.

Como se mostra na revisão bibliográfica anterior não existem modelos matemáticos que descrevam completamente todo os problemas de interesse deste estudo. Porém alguns modelos propostos na literatura são facilmente adaptáveis. Por outro lado, a maioria destes modelos não são satisfatórios na resolução de casos práticos do problema na vida real (BEASLEY, 2004; BEN et al., 2008; CHEN et al., 1995). O enfoque deste trabalho é uma metodologia aproximada, dado que o objetivo é resolver aplicações reais.

Devido a grande quantidade de variantes do problema da mochila consideradas neste trabalho, não são ilustrados os modelos matemáticos das diferentes versões do problema. Na literatura existe uma grande quantidade de formulações propostas e muitas destas são adaptáveis aos problemas indicados neste trabalho, porém nem para todas as variações propostas existe uma formulação matemática na literatura. Isto dificulta uma comparação e uma medição da qualidade da metodologia de solução proposta aqui. Por outro lado, as formulações matemáticas estão fora do alcance deste estudo, pelo que se realizará uma comparação direta com os resultados alcançados com outras metodologias de solução (aproximadas) propostas na literatura.

3.4 METODOLOGIA DE SOLUÇÃO

Devido à diversidade das variantes do problema proposto neste trabalho, tem-se como objetivo definir um esquema geral de otimização para resolver todas as variações propostas, sendo este uma das contribuições principais neste estudo.

O esquema geral de otimização apresentado é baseado na proposta de (ZHU; LIM, 2012) para resolver o problema de carga do contêiner. Aqui adaptamos e propomos características de solução para as especificações do problema da mochila bidimensional. A Figura 36 ilustra os componentes do esquema de otimização geral.

Figura 36 - Esquema geral de otimização proposto para o problema da mochila bidimensional

1. Codificação e/ou representação dos espaços (espaço resultante ou ainda vazio)
2. Construção de blocos (blocos simples ou compostos)
3. Seleção de um dos espaços vazios (definição de critérios de desempate)
4. Seleção de um dos blocos (simples ou compostos) de acordo com a função objetivo
5. Posicionamento do bloco sobre o espaço selecionado
6. Gerenciamento da solução construída
7. Refinamento da solução

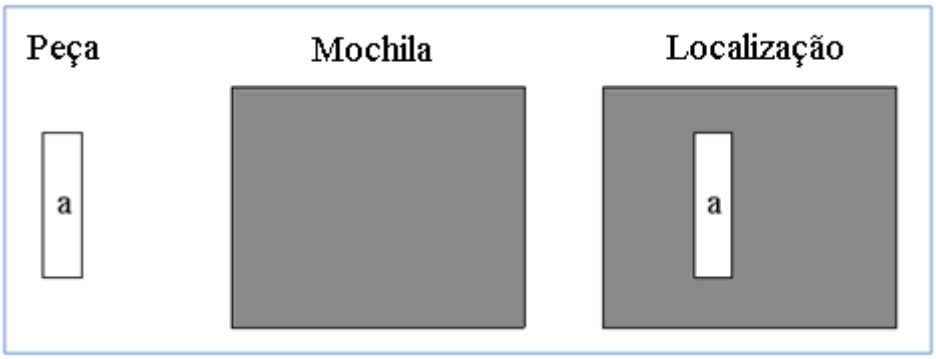
Fonte: Elaboração do próprio autor

As maiorias das metodologias aproximadas apresentadas na literatura seguem implicitamente este esquema de otimização (ao menos parcialmente) ou em geral se encaixam neste esquema. A seguir se descrevem cada um dos componentes do esquema proposto.

Codificação e/ou representação dos espaços

A codificação de espaços consiste em como serão representadas as peças localizadas na mochila e os espaços livres restantes desta (ver Figura 37).

Figura 37 - Peça, mochila e localização de uma sobre a outra



Fonte: Elaboração do próprio autor

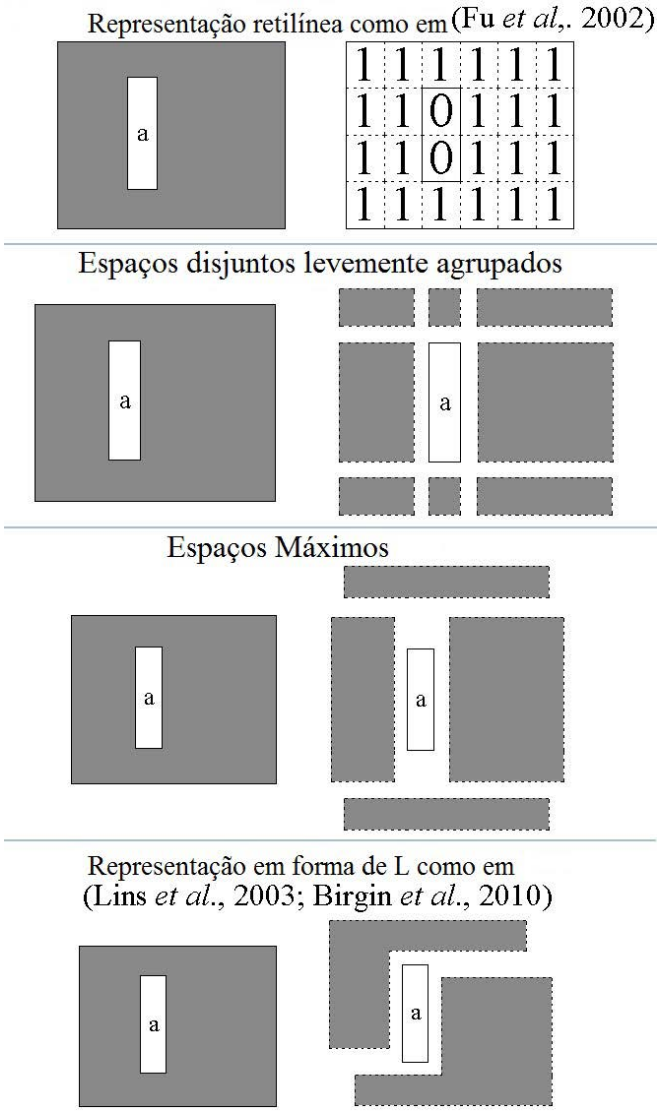
Diferentes representações e codificações têm sido apresentadas na literatura e não existe uma que possa ser chamada da "melhor" em relação às outras, devido que dependendo da codificação escolhida se intentará explorar as características inerentes desta, isto significa que não somente escolher a codificação garante o sucesso da proposta, deve-se zelar por tirar proveito das vantagens e neutralizar as desvantagens do uso desta representação. Na Figura 38 se ilustram algumas codificações (ou representações) utilizadas na literatura.

Construção de blocos (blocos simples ou compostos)

Consiste na criação de estruturas de peças mais complexas a partir das peças originais, isto requer um esforço inicial adicional que é compensado no momento de localizar já não uma peça e sim uma estrutura de peças. Devido que neste trabalho o problema se limita ao estudo de peças regulares, em especial a formas retangulares, as estruturas mais complexas que se podem elaborar são blocos com forma também retangular.

Os blocos são divididos em duas classes: simples e compostos, que se diferenciam no momento de sua fabricação. Os blocos simples consistem em formar estruturas retangulares combinando a demanda de um único tipo de peça. Os blocos compostos se elaboram combinando diferentes tipos de peças que geram uma estrutura similar a um retângulo.

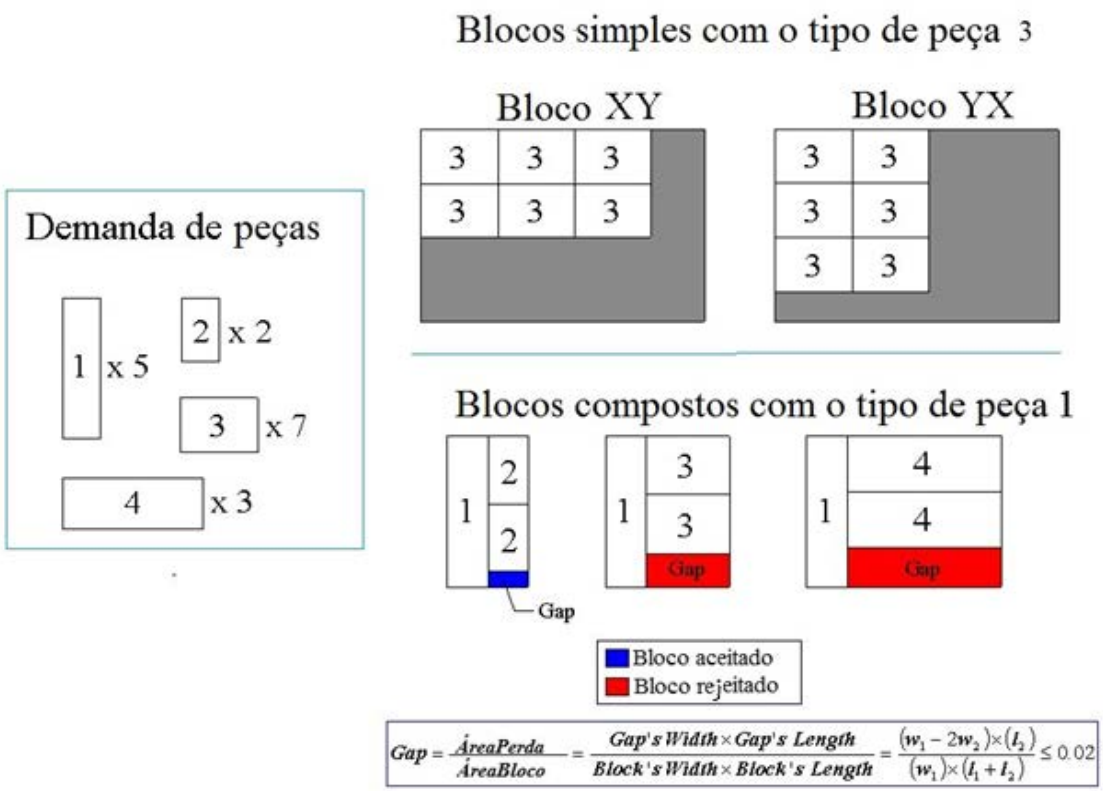
Figura 38 - Representações/codificações da localização das peças sobre os espaços vazios



Fonte: Elaboração do próprio autor

A Figura 39 ilustra a elaboração de blocos simples e compostos, dado um conjunto de peças demandadas. A criação de blocos simples se realiza juntando peças do mesmo tipo, primeiro na direção de uma de suas dimensões (seja comprimento ou largura) e logo sua complementar, isto quer dizer que, podem-se obter no máximo dois blocos simples para cada tipo de peça. Por outro lado, a elaboração de blocos compostos podem-se realizar de diferentes maneiras, usualmente fixar-se um tipo de peça e percorre-se um a um os tipos restantes para formar uma estrutura o mais retangular possível, como não acostuma coincidir as dimensões de unas peças com os múltiplos das outras, deve-se definir um *gap* (uma margem de perda, 2% é recomendado por Zhu et al., (2002)) e relacionar esta informação durante o processo de otimização.

Figura 39 - Elaboração de blocos: simples e compostos



Fonte: Elaboração do próprio autor

Seleção de um dos espaços vazios

Esta etapa consiste em definir os critérios de seleção do espaço vazio a ser preenchido devido que durante o processo se pode ter vários candidatos. Dependendo da ordem em que são escolhidos os espaços se obterão diferentes respostas (soluções ou padrões de corte). Existem diferentes critérios de seleção.

Wu (2002) propõe que a forma mais eficiente é ir preenchendo os espaços mais afastados do centro da mochila (ou seja, escolher os espaços mais próximos das esquinas da mochila) e por último encher os espaços residuais do centro. Para isto se pode definir um índice ou uma distância de medição que comumente se calcula como a distância Euclidiana (ou a distância Manhattan, etc.) entre as esquinas do espaço e as esquinas da mochila original.

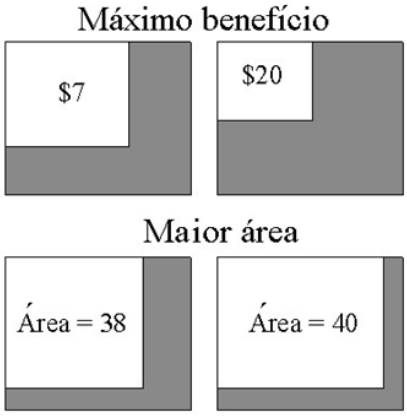
Outro critério de seleção é de acordo com a sua área (neste caso o índice de medição é a área do espaço). Por sentido comum, os espaços menores (de menor área) deveriam ser os primeiros a preencher-se. Adicionalmente, indiferente do critério de seleção, se deve pensar que geralmente ocorrem “empates”, ou seja, mais de um espaço é candidato (tem o melhor índice de medição) a ser o seguinte em preencher-se. Pelo que se deve definir um critério de

desempate, em alguns casos se utiliza o critério de seleção complementar ou também acostuma usar-se uma ordem lexicográfica das dimensões do espaço (primeiro o eixo *X*, segundo o eixo *Y*, último o eixo *Z*).

Seleção de um dos blocos de acordo com a função objetivo

Esta componente consiste em definir uma estratégia de seleção da peça (ou bloco de peças) a localizar-se no espaço vazio selecionado. Na literatura existem diferentes estratégias como: escolher a peça que melhor encaixa (*best-fit*) no espaço vazio, escolher a peça que ocupe maior área (*max area*) do espaço vazio, escolher a peça que produza maior beneficio (*best-profit*), escolher a peça segundo uma medida relativa ou um índice de qualidade especializado, etc. Na Figura 40 ilustram-se duas das estratégias mais usadas na literatura.

Figura 40 - Estratégias de seleção da peça a localizar-se no espaço vazio.



Fonte: Elaboração do próprio autor

Posicionamento do bloco sobre o espaço selecionado

Esta etapa consiste em escolher o lugar exato onde deve ser localizada a peça (ou o bloco de peças) selecionada sobre o espaço vazio. Existem estratégias como fixar uma esquina do espaço vazio onde sempre deve ser localizada a peça (como é realizado na heurística construtiva da esquina inferior esquerda, *bottom-left corner heuristic*) ou escolher uma das esquinas baseado num critério específico (a esquina mais próxima a uma das esquinas da mochila (WU, 2002)).

Gerenciamento da solução construída

As cinco questões anteriores envolvem diferentes cenários devido à presença de diferentes critérios de execução de uma mesma tarefa e decidir sobre qualquer uma delas afetará

diretamente o padrão de empacotamento (a solução) obtido. A maioria das metodologias propostas, geralmente, tomam unicamente decisões deterministas, baseadas nos limites inferiores e superiores ou na previsão dos possíveis padrões (*tree search procedures*). Por outro lado, existem metodologias (algoritmos heurísticos e meta-heurísticos) propostas que acostumam tomar decisões envolvendo um grau de aleatoriedade, isto com a finalidade de oferecer flexibilidade nos padrões de corte (ou empacotamento) que podem ser obtidos.

Figura 41 - Esquema geral de otimização para os problemas de corte

1. Representação dos espaços vazios: ▪ Uso da codificação de árvores de corte para a subdivisão do problema e administração das restrições de tipo de corte • Uso de nós simples com valores reais entre [-1, 1] para as restrições guilhotina. • Uso de nós simples e nós estampados com valores reais entre [0, 1] para as restrições não guilhotina de primeira ordem.
2. Construção de blocos: ▪ Não é gerado nenhum tipo de bloco e durante tudo o processo se manipula a informação peça por peça.
3. Seleção de um dos espaços vazios: ▪ Uso do critério do espaço com menor área.
4. Seleção de um dos blocos: ▪ Cálculo e seleção do melhor bloco dos três critérios: maior área (ou beneficio), melhor ajuste e melhor índice de área (ou beneficio).
5. Posicionamento do bloco sobre o espaço selecionado: ▪ Fixação da esquina inferior esquerda como ponto de referência para a localização de todos os blocos.
6. Gerenciamento da solução construída: ▪ Uso de um algoritmo PSO melhorado para encontrar os valores ótimos da árvore de cortes. ▪ Uso de um algoritmo construtivo de geração de camadas para a localização das peças dentro dos subespaços.
7. Refinamento da solução: ▪ Não é realizado nenhum refinamento sobre as soluções parciais, nem a solução final.

Fonte: Elaboração do próprio autor

Isto significa que, o gerenciamento da solução consiste na definir uma filosofia que relacione cada uno dos critérios selecionados, além de proporcionar ferramentas que forneçem flexibilidade na elaboração dos padrões de corte e empacotamento.

Refinamento da solução

Dependendo da metodologia proposta é possível definir procedimentos que ajudem a refinar ou melhorar a solução obtida.

Esta etapa é comumente aplicada nas metodologias baseadas em meta-heurísticas, onde se podem encontrar rotinas como o algoritmo de Reconexão de Caminhos (*Path Relinking algorithm*) ou aplicar movimentos de melhoramento sobre as soluções obtidas.

Figura 42 - Esquema geral de otimização para os problemas de empacotamento

1. Representação dos espaços vazios: ▪ Uso da codificação de árvores de corte para a subdivisão do problema. • Uso de nós mistos. ▪ Uso dos espaços máximos para a localização direta das peças e blocos de peças sobre os subespaços.
2. Construção de blocos: ▪ Elaboração de blocos compostos, com uma margem de perda (*gap*) do 2%.
3. Seleção de um dos espaços vazios: ▪ Uso do critério do espaço mais próximo a uma das esquinas da mochila.
4. Seleção de um dos blocos: ▪ Cálculo e seleção do melhor bloco dos três critérios: maior área (ou beneficio), melhor encaixa e melhor índice de área (ou beneficio).
5. Posicionamento do bloco sobre o espaço selecionado: ▪ Uso da esquina ganhadora (na seleção de espaços) como ponto de referência para a localização do bloco selecionado.
6. Gerenciamento da solução construída: ▪ Uso de um algoritmo PSO melhorado para encontrar os valores ótimos da árvore de cortes. ▪ Uso de um algoritmo construtivo de geração de camadas para localização das peças dentro dos subespaços.
7. Refinamento da solução: ▪ Não é realizado nenhum refinamento sobre as soluções parciais, nem sobre a solução final.

Fonte: Elaboração do próprio autor

Esquema de otimização proposto

Um dos objetivos deste trabalho era definir um esquema único de otimização para as diferentes variantes do problema da mochila bidimensional. Porém, por causa das diferenças tipológicas que existem entre as variantes do problema se tem definido dois esquemas: um

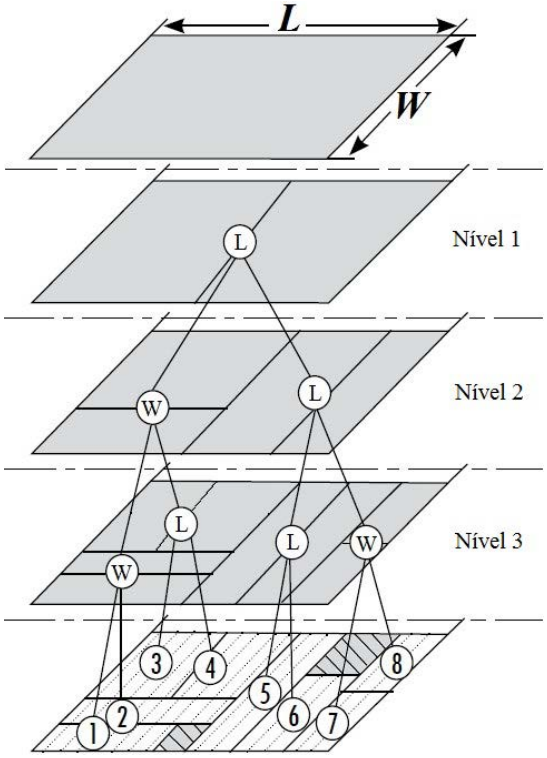
para com os problemas de corte (todas as variantes do problema tipo guilhotina e não guilhotina de primeira ordem) e outro para os problemas de empacotamento (todas as variantes do problema não guilhotina de ordem superior). As Figuras 41 e 42 ilustram o esquema de otimização para os problemas de corte e empacotamento, respectivamente.

A seguir se apresentam um a um os elementos utilizados no esquema de otimização proposto.

3.4.1 Codificação proposta

Wong et al. (1988) apresentam uma estrutura de codificação para o problema de desenho de planta (VLSI, *Very Large Scale Integration*) denominada árvore de cortes. Uma das grandes vantagens da representação em árvore de cortes é a geração de padrões de corte tipo guilhotina e não guilhotina de primeira ordem. Diferentes metodologias propostas têm corroborado na efetividade da codificação em árvore de cortes, em especial as apresentadas por (KRÖGER, 1995), (CUI, 2007a; 2007b) e (TORO et al., 2008).

Figura 43 - Representação de cortes e disposição das peças através de uma árvore de cortes

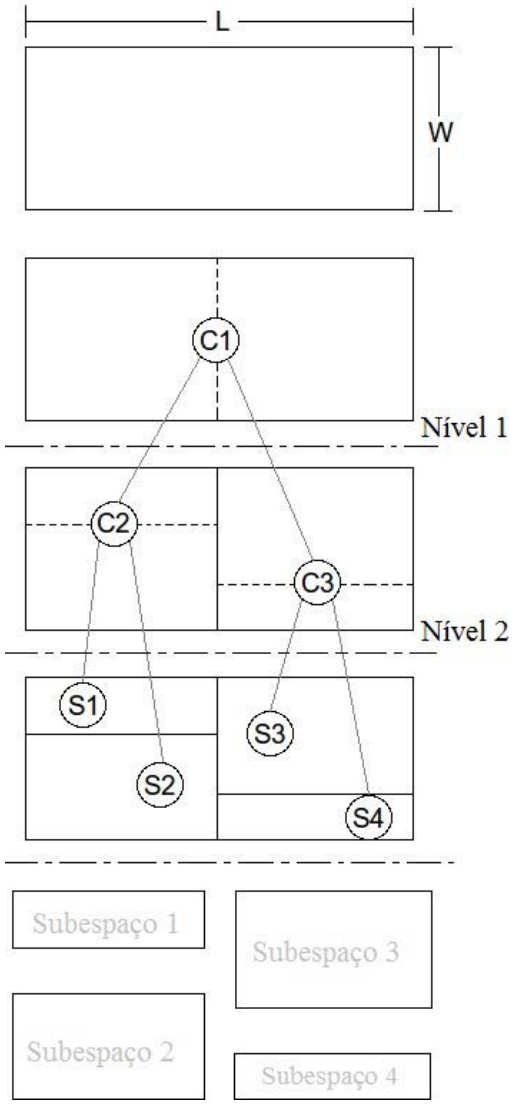


Fonte: Elaboração do próprio autor

Uma árvore de cortes se define como uma árvore com raiz, onde cada nó interno (nó pai) representa a posição e a forma como se realiza o corte sobre o material (horizontal ou vertical), enquanto os nós folha (nós terminais) representam as dimensões dos subespaços gerados para cortar as peças agrupadas (ver Figura 43).

Na Figura 44 ilustra-se uma árvore de cortes para o problema, em que o nó raiz (nível 1) indica como e onde um corte perpendicular ao comprimento da mochila se deve realizar. A hierarquia da árvore denota que o nó (filho) esquerdo representa um corte perpendicular à largura do subespaço resultante da esquerda, e o nó (filho) direito faz um novo corte perpendicular à largura do subespaço resultante da direita, isto para o segundo nível. Depois, no terceiro nível são obtidos os subespaços 1 e 2 do lado esquerdo e os subespaços 3 e 4 do lado direito. Em cada subespaço gerado serão localizadas por camadas as peças.

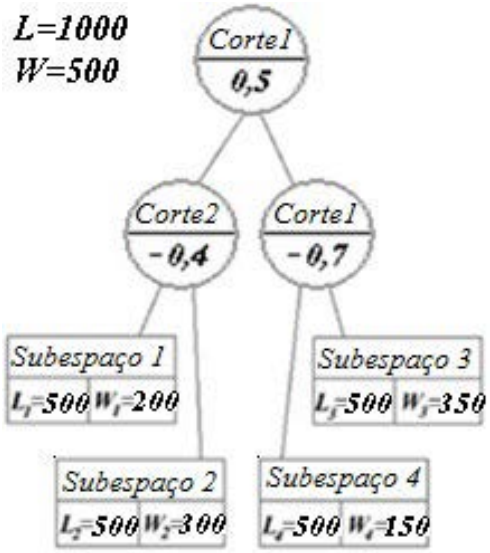
Figura 44 - Árvore de cortes tipo guilhotina de dois níveis



Fonte: Elaboração do próprio autor

Portanto, a árvore de cortes contém em seus nós internos a informação sobre a orientação dos cortes (seja perpendicular ao comprimento ou à largura) e a distância a qual o corte deve ser realizado. Por outro lado, os nós folhas contêm as dimensões dos subespaços resultantes.

Figura 45 - Exemplo de uma árvore de cortes



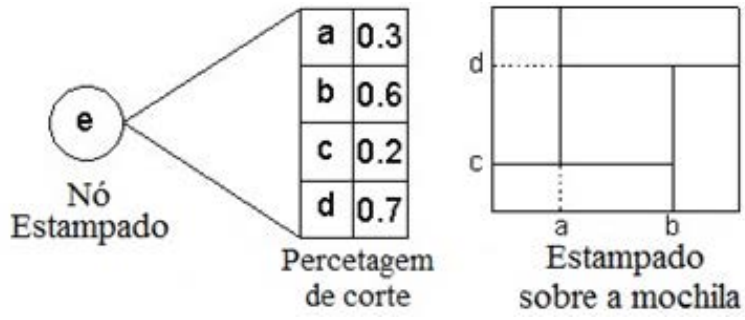
Fonte: Elaboração do próprio autor

Como exemplo, na Figura 45 apresenta-se uma árvore de cortes sobre uma mochila de 1000 unidades de comprimento e 500 unidades de largura. As dimensões dos espaços resultantes são apresentadas na Tabela 1. Note que cada nó da árvore de corte, exceto as folhas, contém a orientação (representado através do sinal, positivo se é perpendicular ao comprimento e negativo se é perpendicular à largura) e a distância dos cortes (representada através de um número do 0 ao 1, onde este é a proporção entre espaços e largura). Portanto, o nó raiz indica que um corte perpendicular no comprimento ao 50% da mochila deve ser realizado, dividindo esta em dois novos subespaços (note que este foi um corte tipo guilhotina) ambos de dimensões iguais, 500 unidades de comprimento e 500 unidades de largura. Assim sucessivamente, a árvore guia o processo de corte e nas folhas finalmente se armazenarão os subespaços.

Tabela 1 - Dimensões dos subespaços

Subespaço	**Comprimento**	**Largura**
1	500	200
2	500	300
3	500	350
4	500	150

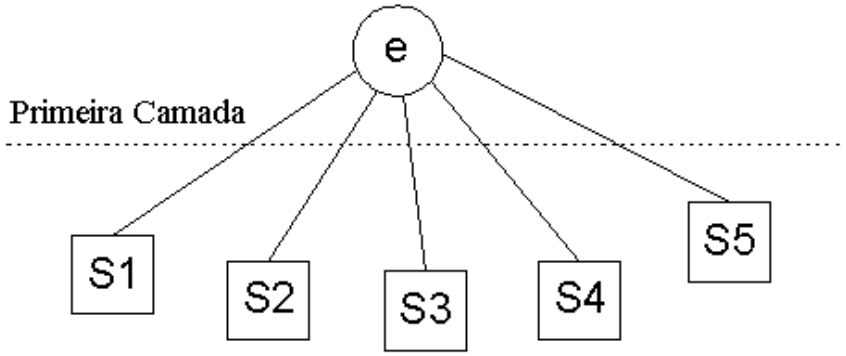
Figura 46 - Nó estampado não guilhotina sobre a mochila



Fonte: Elaboração do próprio autor

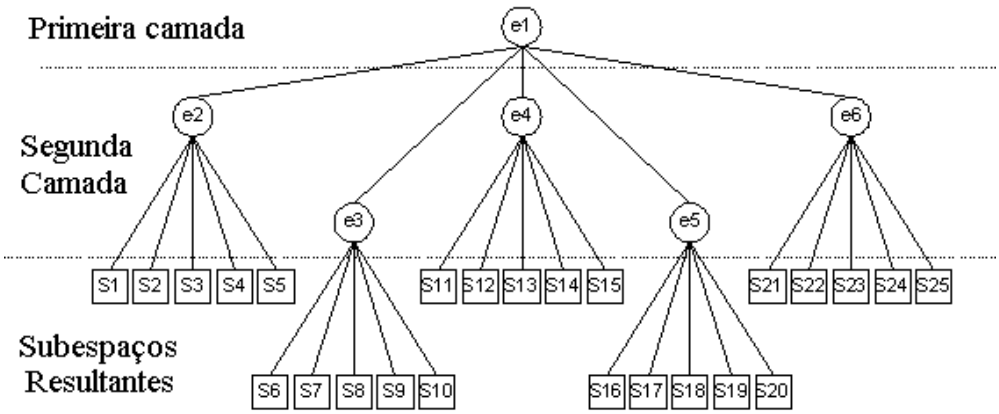
Por outro lado, para garantir que os subespaços criados apresentem cortes de tipo não guilhotina se define um nó estampado (quatro cortes que geram cinco subespaços, ver Figura 46). Para representar a estrutura que gera os subespaços se define uma árvore mista (entre 5-ário e binário) onde um nó pai (nó interno) pode representar um nó estampado ou um nó guilhotina, enquanto os nós folhas continuam representando as dimensões dos subespaços resultantes.

Figura 47 - Árvore de cortes com um nó estampado, árvore de primeiro nível



Fonte: Elaboração do próprio autor

Figura 48 - Árvore de cortes de duas camadas (dois níveis)

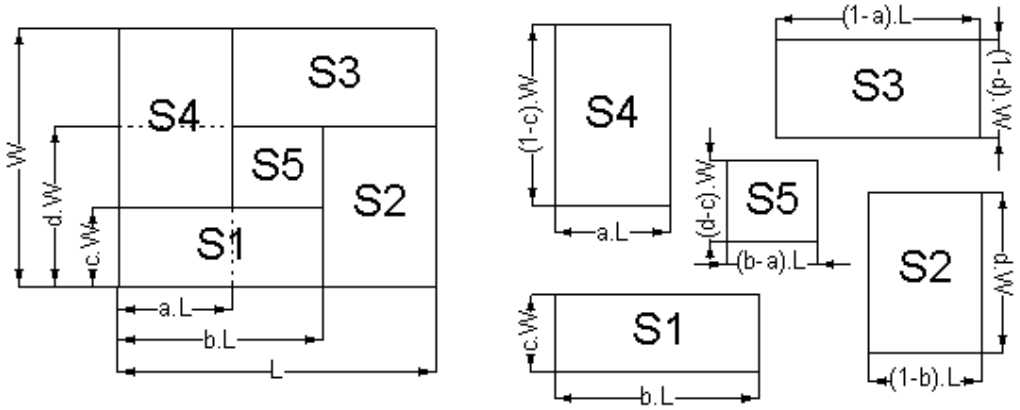


Fonte: Elaboração do próprio autor

Por exemplo, a Figura 47 (primeiro nível) ilustra uma árvore de cortes. Uma vantagem deste tipo de codificação é que qualquer conjunto de valores da árvore produz um padrão de corte factível. A Figura 48 ilustra uma árvore de cortes de duas camadas com 6 nós estampados e 25 subespaços resultantes. A Figura 49 ilustra as dimensões dos subespaços resultantes.

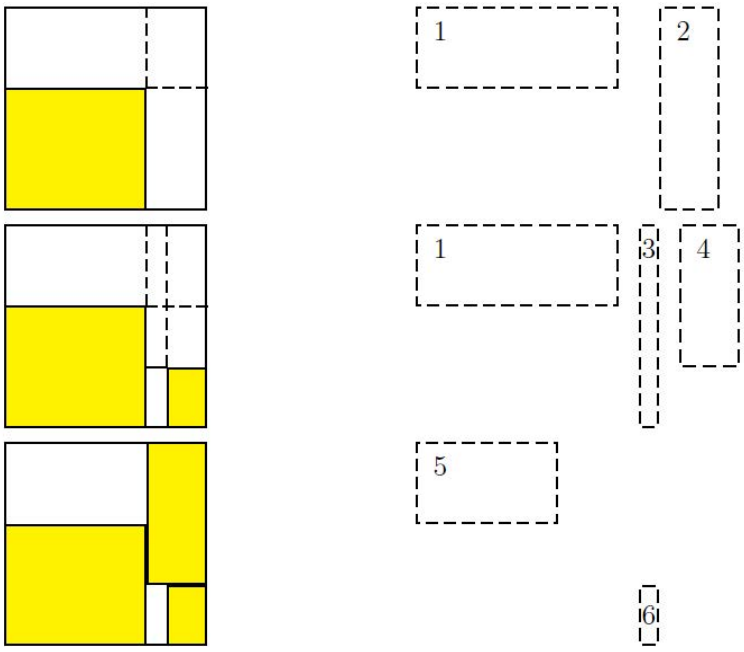
Depois de obter as dimensões dos subespaços se passa ao passo à localização das peças. Para isto se utiliza uma heurística de criação de camadas, a qual utiliza uma representação do espaço livre chamada de “espaços máximos” não disjuntos (em ordem para conservar as restrições de corte tipo guilhotina e corte tipo não guilhotina de primeira ordem se utiliza uma representação de espaços máximos residuais disjuntos).

Figura 49 - Dimensões resultantes dos cinco espaços, através de quatro cortes (nó estampado)



Fonte: Elaboração do próprio autor

Figura 50 - Espaços máximos



Fonte: Parreño et al. (2008)

3.4.2 Representação do espaço livre: espaços máximos residuais

Um espaço máximo, como seu nome o indica, consiste em uma submochila retangular do máximo tamanho possível, reconhecível no espaço livre restante da mochila depois de localizar uma peça (ou uma camada destas). Na Figura 50 se mostram os espaços máximos na mochila que estão sendo preenchidos e logo atualizados ou eliminados. Deve-se notar que estes são não disjuntos, e para as versões de corte tipo guilhotina e não guilhotina de primeira ordem os espaços "máximos" selecionados e definidos deverão ser disjuntos.

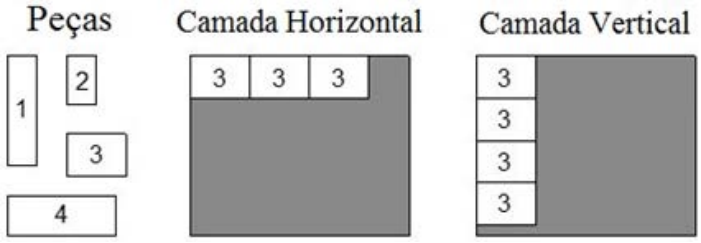
3.4.3 Algoritmo heurístico construtivo de criação de camadas

O algoritmo construtivo proposto por Parreño et al. (2008), cuja estratégia de empacotamento consiste em criar camadas de peças a partir de uma esquina selecionada. Na Figura 51 se mostra como se constrói uma camada a partir de alguma das peças disponíveis.

A especificação do algoritmo construtivo é a seguinte:

1) Identificam-se os espaços máximos disponíveis na mochila e se escolhe um dependendo do critério esquinas selecionado. Em caso de empate, seleciona-se o subespaço com menor área.

Figura 51 - Camadas vertical e horizontal geradas por alguma das peças que será cortada ou empacotada



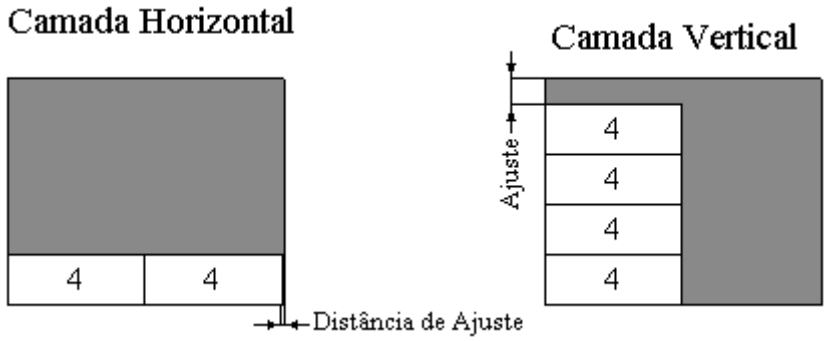
Fonte: Elaboração do próprio autor

2) Uma vez escolhido o subespaço se gera uma camada vertical e uma horizontal para cada um dos tipos de peças que se vão cortar ou empacotar. Escolhe-se a peça que apresente a melhor medida dos seguintes critérios:
   a) Máximo benefício: Seleciona-se a peça que gere a camada (seja vertical ou horizontal) o maior benefício. Na versão com valores de peças não ponderados a configuração (a camada de peças) selecionada se mede com base na área que ocupa.

b) Melhor ajuste: Seleciona-se a camada que fica mais perto dos limites do subespaço máximo que se está preenchendo. Na Figura 52 se ilustra o melhor ajuste.

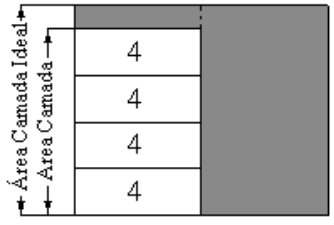
c) Índice área: Neste trabalho se propõe outro critério para selecionar as peças que se adicionarão, denominado *índice área*, que consiste em relacionar a área ocupada pela camada que produz uma peça com o área da camada ideal que seria equivalente a uma camada com distância de ajuste 0. Na Figura 53 se ilustra a relação.

Figura 52 – Camada melhor ajuste. Se a distância entre a camada e o limite do subespaço é menor, diz-se que apresenta um melhor ajuste (*best-fit*)



Fonte: Elaboração do próprio autor

Figura 53 – Camada melhor índice. Quando o índice é maior, a camada gerada apresenta um maior aproveitamento do espaço delimitado pela linha descontínua denominada camada ideal.



Índice Área = Área Camada / Área Camada Ideal

Fonte: Elaboração do próprio autor

3) Uma vez eleita a peça e a orientação da camada, se atualizam os subespaços máximos e voltamos ao passo 1 enquanto existam espaços máximos livres.

Deve-se notar que, para com os problemas com restrições de corte tipo guilhotina por etapas, somente é permitida a criação de camadas num único sentido (vertical ou horizontal), o qual deve corresponder à orientação do nó de corte que produziu o subespaço.

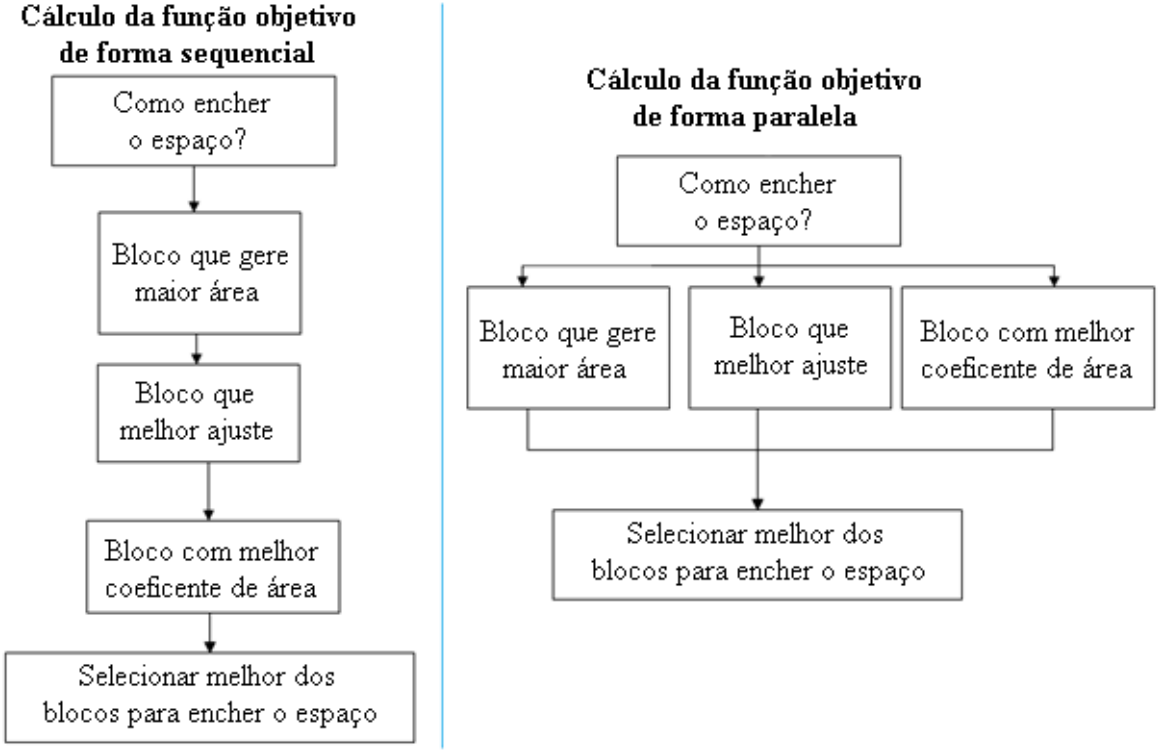
3.4.4 Cálculo da função objetivo

Nos subespaços gerados pela árvore de cortes se deve realizar a localização das peças, este processo se realiza através do algoritmo heurístico construtivo utilizando a representação de espaços máximos livres, garantindo os padrões de corte e à vez que se cortem a maior quantidade de peças por subespaço (para a versão onde as peças têm valor não ponderado) ou o conjunto de peças que gere melhor beneficio (para a versão onde as peças têm valor ponderado).

O algoritmo heurístico com os três diferentes critérios mencionados, localizará camadas de peças nos subespaços. A função objetivo será calculada somando a área ocupada ou o custo associado das peças localizadas em todos os subespaços na que foi dividida a mochila original.

Os diferentes critérios utilizados para decidir a forma de preencher estes subespaços em cada iteração representam diferentes esforços computacionais. Aplicá-los simultaneamente gera um gargalo notável especialmente quando é aumentado o número de iterações no contexto da meta-heurística. Por esta razão neste trabalho se utiliza o recurso computacional oferecido pelo processamento paralelo (ver Figura 54).

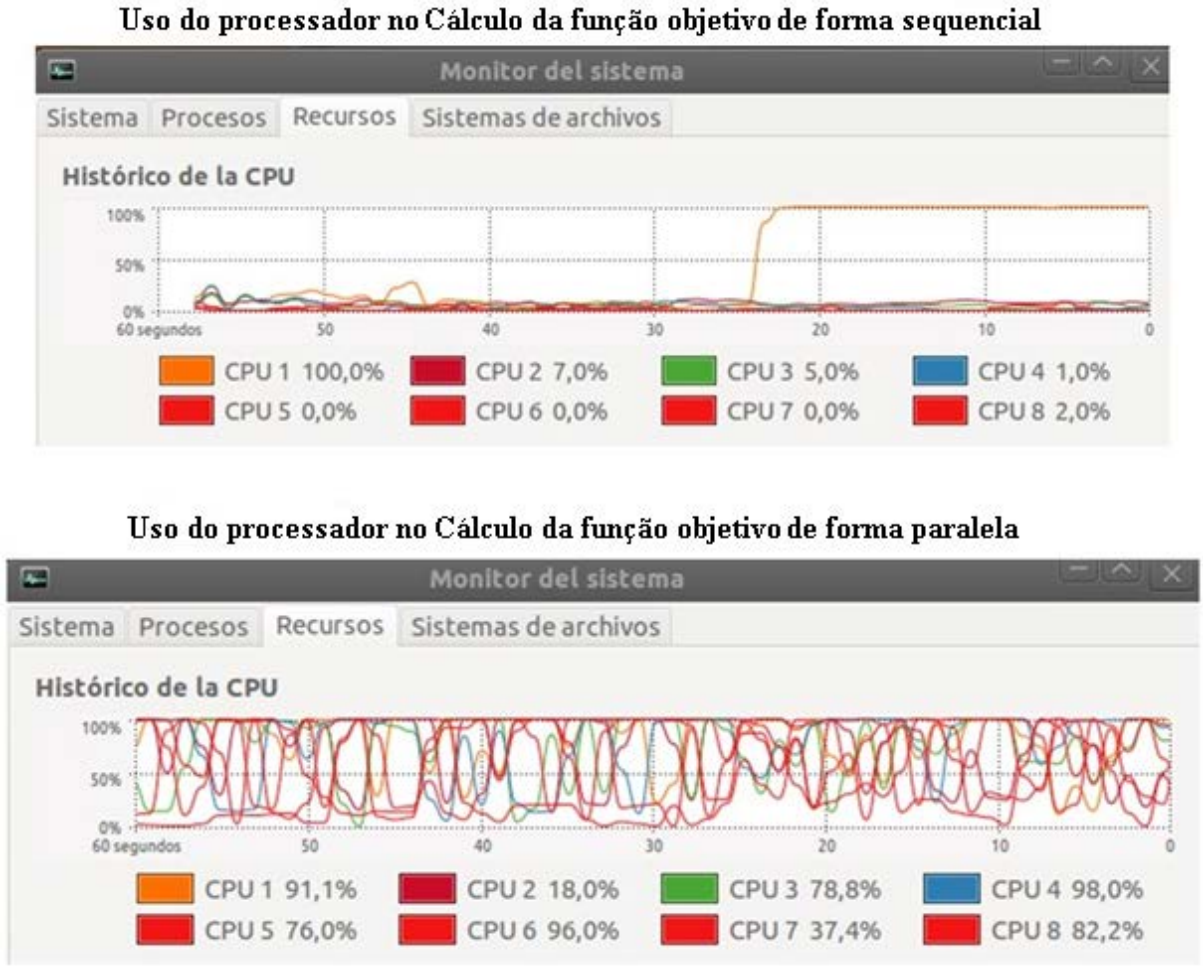
Figura 54 - Cálculo da função objetivo, gargalo sequencial, paralelização do gargalo



Fonte: Elaboração do próprio autor

O processamento paralelo é uma técnica que permite a realização de múltiplas tarefas de maneira simultânea, seja aproveitando momentos de inatividade de somente um processador para conseguir esta simultaneidade ou integrando vários processadores para realizar diferentes tarefas no mesmo instante de tempo. Esta forma de execução se pode realizar em um ou muitos processadores de uma mesma máquina (usando OpenMP, biblioteca de funções do C++ para este propósito) ou em um ou vários processadores distribuídos em diferentes máquinas interconectadas através de uma rede (utilizando MPI igualmente para C++). Para o trabalho desenvolvido se escolheu a paralelização de tarefas numa máquina só, dado que se contou com um computador com vários núcleos (como neste caso que estava disponível um processador i7®) se apresenta um melhor rendimento ao evitar a latência inerente ás redes de dados que comunicam processadores distribuídos em diferentes máquinas (ÁLVAREZ et al., 2011).

Figura 55 - Uso do processador das diferentes versões: cálculo sequencial e cálculo em paralelo.



Fonte: Elaboração do próprio autor

Esta simultaneidade é possível graças a biblioteca OpenMP que permite dividir um único curso de eventos em vários cursos simultâneos, denominados fios de execução. Estes fios de execução se executam de maneira síncrona em nível de tarefas (todos iniciam e terminam ao mesmo tempo) e de forma assíncrona em nível de bits (iniciam e terminam em momentos diferentes) de maneira automática pela biblioteca OpenMP (ver Figura 55).

A codificação proposta neste estudo garante a factibilidade quanto às restrições de padrões de corte. Portanto, é necessário encontrar os valores ótimos dos cortes que dividem a mochila, para isto se propõe o algoritmo da próxima subseção.

3.4.5 Algoritmo otimização por enxame de partículas, busca em vizinhança variável e algoritmos genéticos ($A_{PSO+VND+TF}$)

O algoritmo proposto neste trabalho utiliza diferentes técnicas de otimização, combinando características de otimização por enxame de partículas (PSO) com busca em vizinhança variável (VND), e o operador de mutação dos algoritmos genéticos. O primeiro é considerado o algoritmo principal, o segundo foi utilizado como um limitador das características das partículas que devem ser atualizadas, e o conceito da mutação foi utilizado como o fator de turbulência de voo das partículas (ÁLVAREZ et al., 2010).

O algoritmo PSO é uma metaheurística pertencente ao grupo dos algoritmos evolutivos e sua estratégia se fundamenta no comportamento social dos animais tais como as bandos de aves, cardume de peixes ou enxames de abelhas, os quais atuam como se fossem um indivíduo só. Nestes grupos de animais se estabelecem relações entre os indivíduos, se criam hierarquias dependendo das características dos mesmos, existindo um líder grupal reconhecido e seguido pelos demais membros do grupo. O papel de líder pode trocar se existe outro indivíduo com melhores características que o líder existente. Quando o grupo se organiza para realizar uma tarefa, o líder guia os demais indivíduos para uma região promissora, porém, os demais membros do grupo durante seu percurso podem inferir sobre uma nova direção com o objetivo de ter um melhor sucesso na missão.

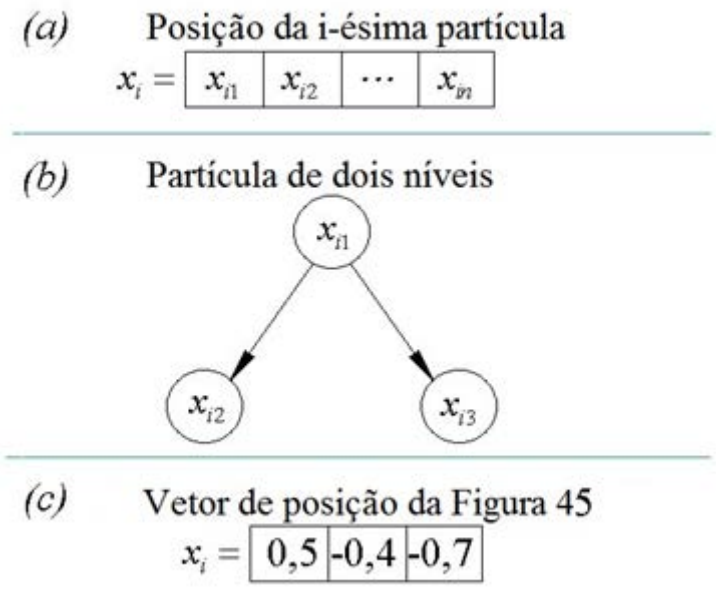
Em PSO a exploração do espaço de solução se realiza através de uma população de indivíduos conhecidos como partículas, onde cada uma delas representa uma possível solução do problema. A localização de cada partícula sobre a região de busca está determinada mediante sua posição, a qual, representa o valor que tomam as variáveis de decisão do problema (KENNEDY; EBERHART, 2001).

Cada partícula muda de posição de acordo a sua velocidade levando em conta a melhor solução encontrada por esta durante o processo (*pbest*) e a informação do líder do enxame (*gbest*). O operador *pbest* (*individual best*) compara a posição atual da partícula com sua melhor posição que foi encontrada na sua memória. Enquanto que o operador *gbest* (*global best*) estuda o comportamento do grupo, armazenando a posição do líder atual do enxame (ver Figura 58).

População inicial

As técnicas heurísticas, dependendo da complexidade matemática do problema, geram a população inicial de forma aleatória ou utilizando métodos construtivos baseados em fatores de sensibilidade, e sua finalidade consiste em iniciar a busca numa região atrativa, isto para diminuir o tempo e esforço computacional. O caso de estudo apresentado neste trabalho inicia com um conjunto de partículas geradas de forma aleatória. Porém, na medida em que se estudam sistemas grandes e de alta complexidade matemática ganha força o uso de uma população inicial gerada com base nos métodos construtivos.

Figura 56. Representação da *i*-ésima partícula.



Fonte: Elaboração do próprio autor

(a) Vetor Posição da *i*-ésima partícula.

(b) Exemplo de uma partícula que representa uma árvore de dois níveis.

(c) Vetor posição (notação polonesa) da árvore de cortes ilustrada na Figura 45.

A população é constituída por *k* partículas e cada uma destas no estado *i* representa uma alternativa de solução, que é interpretada no problema através da posição da partícula. Cada partícula se representa mediante um vetor, os valores nas posições deste indicam os valores das variáveis do problema (ver Figura 56a) e simbolizam um ponto dentro do espaço de solução. Inicialmente se geram *n* partículas com valores aleatórios dentro do intervalo das variáveis e através da função objetivo se determina a qualidade da solução.

Neste trabalho cada partícula representa uma árvore de cortes (ver Figura 56b), para melhor abstração neste documento usamos uma representação vetorial das variáveis: posição e velocidade da partícula. Estes vetores podem entender-se como a notação polonesa das árvores de cortes utilizados (ver Figura 56c).

Seleção do líder

Em PSO, durante cada iteração, deve selecionar-se o líder do grupo comparando os valores da função objetivo de cada partícula com a função objetivo do líder, sendo este último aquela partícula que tem o melhor valor (incumbente global). Durante o processo de otimização, cada vez que uma partícula melhore o valor de sua função objetivo, deve-se atualizar o valor de seu melhor localização local (*pbest*) e cada vez que exista uma troca de líder, deve-se atualizar o valor da melhor localização global (*gbest*).

Função de velocidade

A velocidade permite atualizar a posição de cada partícula. O vetor de velocidade representa o gradiente de cada indivíduo dentro do enxame, ou seja, guia às partículas durante o processo de busca.

Figura 57. Velocidade da *i*-ésima partícula.

$$v_i = \boxed{v_{i1}} \boxed{v_{i2}} \boxed{\cdots} \boxed{v_{in}}$$

Fonte: Elaboração do próprio autor

Da mesma forma que a posição, a velocidade se representa através de um vetor aonde suas dimensões devem ser iguais (ver Figura 57). A velocidade contém informação da experiência local e coletiva do grupo. Para calcular a velocidade de cada partícula se usa a Equação 3.

$$v_i(t+1) = w \cdot v_{i-1}(t) + c_1 \cdot rand \cdot \left(pbest - x_i(t) \right) + c_2 \cdot Rand \cdot \left(gbest - x_i(t) \right) \quad (3)$$

Onde os elementos usados são:

Velocidade atual (v_{i-1}): Direção de voo que apresenta uma partícula. Assume-se que no começo do processo as partículas partem do repouso.

Fator de inércia (w): Fator de ponderação que atua sobre a velocidade atual da partícula. Um valor elevado representará continuar com a trajetória que vem traçando a partícula, enquanto que um valor pequeno significará liberdade de tomar novas trajetórias.

Constantes de aceleração c_1 e c_2: Valores que direcionam as partículas para sua melhor localização local e global, respectivamente. A adequada calibração destes valores permite que a população não se homogenize durante o processo (SHI; EBERHART, 1998).

Rand e *rand*: Valores aleatórias pertencentes a uma distribuição de probabilidade uniforme.

Atualização da posição

Para determinar a nova posição das partículas se aplica a Equação 4.

$x_i(t+1) = x_i(t) + v_i(t+1)$	*(4)*

Como a posição atualizada das partículas deve satisfazer as restrições das variáveis do problema, sendo necessário definir um intervalo de valores para as velocidades, evitando assim ultrapassar os limites impostos. Além disto, para as restrições de corte tipo guilhotina de duas etapas, os nós de corte sem filhos deverão conservar a direção de corte de seus nós pais.

Fator de turbulência de voo

Em PSO a velocidade da partícula *i* é dada por $V_i = (v_{i1}, \ldots, v_{iN})$ a qual é atualizada através da Equação 4. Durante o desenvolvimento inicial do PSO (KENNEDY; EBERHART, 1995), uma variável estocástica chamada de "loucura" se utilizou para alterar as partículas, desta forma se adicionava a Equação 5:

$v_{i,j} := v_{i,j}^{\phi} + r_3$	*(5)*

Onde $v_{i,j}^{\phi}$ é a velocidade da *j-ésima* característica da partícula X_i e r_3 é a variável de loucura aleatória. O parâmetro de loucura é utilizado para manter a diversidade na população. Neste trabalho se introduz esta variável dentro do PSO através do operador mutação, pertencente aos algoritmos genéticos. Para manter a concepção do PSO, este termo é referido como uma turbulência (equivalente a uma perturbação), porque reflete uma mudança no voo das partículas quando ficam fora de controle.

Figura 58 - Algoritmo PSO

Inicio

t = 1; *população* = GeraçãoPopulaçãoInicial(*tamanhopopulação*)

gbest = SelecionarLíder(*população*)

Se *t = 1*

$pbest_i = X_i$

$gbest = max(pbest_i)$

Caso contrário

Se f.o.($D_i(t)$) < f.o.($pbest_i$)

$pbest_i = X_i$

Se f.o.($D_i(t)$) < f.o.(*gbest*)

$pbest_i = X_i$

Fim Se

Fim Se

Fim Se

velocidade = CalcularVelocidade(*população,pbest,gbest,*c_1*,*c_2*,w*)

população = AtualizarPosição(*população,velocidade*)

t = t + 1

Se o critério de parada foi satisfeito

Finaliza-se o algoritmo

Caso contrário

Voltar à etapa de Seleção do Líder

Fim Se

Fim

Fonte: Elaboração do próprio autor

Critério de parada

Os algoritmos de otimização precisam de um critério que lhes permita decidir quando finalizar a exploração do espaço de solução. Entre os critérios de parada mais usados nesta classe de algoritmos se encontram:

- Se durante um número de iterações não há melhorado a incumbente, escolhe-se esta solução e se dá por finalizado o processo.

- Até cumprir um número máximo de iterações.

Esta última opção é aplicada neste trabalho.

Descrição do algoritmo

O procedimento utilizado pelo algoritmo PSO é apresentado na Figura 58.

Hibridação do algoritmo

Neste trabalho o VND (V*ariable Neighborhood Descent*, proposto por Mladenović e Hansen, 1997) é incluído dentro do PSO com o propósito de definir quais características devem ser atualizadas pelas Equações 3 e 4. Portanto, o conjunto de vizinhanças N_i^l (donde, $\boldsymbol{i = 1, 2, \ldots, Nds}$, sendo *Nds* o número total de nós internos da arvore de corte) é definido como o número *i* de características que devem mudar de valor na partícula *l*.

i=1, então, N_1 = {uma característica aleatória deve mudar seu valor}

i=2, então, N_2 = {duas características aleatórias devem mudar seus valores}

e assim generalizando a *k-ésima* vizinhança seria.

i=k, então, N_k = {*k* características aleatórias devem mudar seus valores}

Portanto, o conjunto resultante de vizinhanças é $\boldsymbol{N} = \{\boldsymbol{N}_1, \boldsymbol{N}_2, \cdots, \boldsymbol{N}_{Nds}\}$,

O operador de mutação comumente usado nos algoritmos genéticos é introduzido no algoritmo proposto tentando simular o fator de turbulência de voo. Neste trabalho se apresenta uma adaptação da Equação para atualizar a variável limiar do algoritmo de otimização Aceitando o limiar (*Threshold Accepting Algorithm*, DUECK; SCHEUER, 1990).

O mecanismo de turbulência consiste em permitir grandes mudanças durante as primeiras iterações, da mesma forma que o algoritmo aceitando o limiar permite a perda de qualidade para com a função objetivo no começo do processo (devido à relaxação do limiar). Com o avanço do processo, a turbulência será mais determinista. O fator de turbulência é definido como a modificação do valor do nó da árvore, através do mecanismo mostrado na Equação 6.

$$node\ i = node\ i + \left(rand - \frac{1}{2} \right) \cdot \left(\sqrt{1 - \frac{k}{TotalIterations}} + \varepsilon \right) \qquad (6)$$

A Equação 6 é composta pelo valor atual do nó *i* da árvore, *rand* é um número aleatório com distribuição uniforme no intervalo $[0, 1]$, *k* é a iteração atual, *TotalIterations* é o número total

de iterações e ε é a percentagem mínima para gerar uma troca no valor da árvore, onde $\varepsilon = 100/\max(L,W)$. Neste trabalho se introduz o operador de mutação apresentado nos algoritmos genéticos, utilizando o parâmetro chamado de taxa de mutação, permitindo a mutação de partículas em cada iteração.

Figura 59 - Algoritmo híbrido de PSO, VND e Fator de Turbulência ($A_{PSO+VND+TF}$)

Inicio
Inicializar população e parâmetros
Para *l = 1* até *NúmeroTotalIterações*
Para *i = 1* até *TamanhoPopulação*
Se $f(\overline{x_i}) < f(\overline{pbest_i})$, **então**

$\overline{pbest_i} = \overline{x_i}$

$N^i = N_1$
Se $f(\overline{x_i}) < f(\overline{gbest})$, **então**
$gbest = \overline{x_i}$
Fim Se
Caso contrário

$\boldsymbol{N^i = Seguinte\ \ N^i}$

Fim Se
Para $\boldsymbol{k \in N_i}$

$v_i^k = wv_{i-1}^k + c_1 rand_1^k \left(pbest_i^k - x_i^k\right) + c_2 rand_2^k \left(gbest_i^k - x_i^k\right)$

$x_i^k = x_{i-1}^k + v_i^k$

Seguinte *k*
Se *rand* < Taxa de Mutação
$\boldsymbol{c\_rand = \lfloor rand \cdot Nds \rfloor}$

$x_i^{c\_rand} = x_i^{c\_rand} + \left(rand - \frac{1}{2}\right) \cdot \left(\sqrt{1 - \frac{l}{NúmeroTotalCiclos}} + \varepsilon\right)$

Fim Se
Seguinte *i*
Seguinte *l*
Fim

Fonte: Elaboração do próprio autor

Dado que o algoritmo PSO apresenta algumas similaridades com os algoritmos evolutivos como os algoritmos genéticos, diferentes autores já propuserem a inclusão do operador de mutação no algoritmo PSO.

Estas operações híbridas normalmente se programam em cada geração (ANDREWS, 2006), (CARLISLE; DOZIER, 2000) ou num intervalo prefixado (LIANG; SUGANTHAN, 2005) ou são controladas por uma função de adaptação definida. Neste estudo, em cada geração do algoritmo PSO, a população tem a probabilidade de utilizar o operador de mutação.

A Figura 59 mostra o pseudocódigo do algoritmo $A_{PSO+VND+TF}$, neste esquema cada partícula entra em um loop de atualização que está condicionado pela vizinhança da partícula, limitando desta maneira o número de cortes que serão atualizados, além disto, o operador de mutação é introduzido e será realizada uma perturbação na posição da partícula se e somente se o valor aleatório é menor que a taxa de mutação.

Tabela 2 - Valores dos parâmetros do algoritmo $A_{PSO+VND+TF}$

Parâmetro	**Valor**
Tamanho da população	100
Número de iterações totais	100
c1(Conhecimento Individual)	2.05
c2(Conhecimento Coletivo)	2.05
w(Inércia)	0.6
Número de níveis da árvore de cortes	3
Taxa de mutação	0.03

Calibração de parâmetros

O ajuste ótimo dos parâmetros é de grande relevância no processo de implementação de uma técnica meta-heurística porque deste depende diretamente a qualidade das respostas encontradas na solução de um problema específico. Não existe um método exato, eficiente e

geral para realizar a calibração dos parâmetros das diferentes técnicas meta-heurísticas, comumente estes algoritmos são calibrados através da combinação de uma busca exaustiva e uma análise estatística da qualidade dos resultados.

Zhan et al. (2009) apresentam um intervalor de valores reduzido para os parâmetros do algoritmo PSO, o qual reduz consideravelmente o espaço de busca dos melhor valores. Adicionalmente, neste trabalho se conserva a filosofia do operador de mutação dos algoritmos genéticos, onde a probabilidade de que aconteça uma mutação na população é muito pequena.

Na Tabela 2 mostram-se os valores dos parâmetros utilizados neste trabalho, estes são tomados do estudo apresentado em (ÁLVAREZ et al., 2011a). Neste é selecionado uma instância de teste aleatória de cada variante do problema, obtendo assim um conjunto de 20 instâncias, é fixado a semente dos valores aleatórios do algoritmo, os intervalos dos parâmetros são divididos como sugere Álvarez et al. (2011a) e são executadas todas as diferentes combinações, selecionando assim a combinação de parâmetros com o maior número de melhores soluções alcançadas. Desta forma se obtêm uns valores únicos dos parâmetros do algoritmo híbrido proposto.

3.5. Análise de resultados

As instâncias de teste usados neste estudo foram tomados da literatura especializada, metodologias aproximadas e exatas são usadas na solução destes problemas. Os problemas selecionados são variados quanto à complexidade matemática e foram propostos especialmente para cada tipo de problema.

No primeiro conjunto de instâncias para os problemas com padrões de corte tipo guilhotina, sem um número de etapas de corte e com um número de exemplares restrito (*Constrained version*) foram selecionados vinte e seis (26) casos. Quinze (15) destes para a versão de peças com valores ponderados: CHW1 e CHW2 tomado de (CHRISTOFIDES; WHITLOCK, 1977), TH1 e TH2 de (TSCHÖKE; HOLTHÖFER, 1996) e o grupo de instancias de CW1-CW11 tomadas de (HIFI, 1997a). Onze casos para a versão de peças com valores não ponderados, CU1- CU11 tomados de (HIFI, 1997b). Estes 26 casos apresentam diferentes tipos de mochilas com distribuições entre 25 e 50 tipos de peças, todos estes pertencem à categoria de problemas de corte com média e alta complexidade.

Sessenta e três (63) casos de prova foram selecionados para os problemas com duas etapas de corte, sendo que cinquenta (50) casos para a versão de peças com valores não ponderados, 2SCUI1- 2SCUI50 tomados de (CUI, 2007a) e treze (13) casos para a versão de peças com valores ponderados, tomados de (HIFI; ROUCAIROL, 2001). Estes 63 casos apresentam diferentes tipos de mochila com distribuições entre 5 e 40 tipos de peças, e estes pertencem à categoria de problemas de corte de pequena e média complexidade.

Quarenta (40) instâncias foram selecionadas para os problemas com três etapas de cortes, sendo que vinte (20) instâncias para a versão de peças com valores não ponderados e vinte (20) instâncias para a versão com valores ponderados, 3SCUI1- 3SCUI20 tomados de (CUI, 2008). Estas 40 instâncias apresentam diferentes tipos de mochila com distribuições entre 40 e 100 tipos de peças, estes pertencem à categoria de problemas de corte de alta complexidade.

Por outro lado, para os problemas com padrões de corte tipo guilhotina, sem um número de etapas de corte e com um número de exemplares irrestrito (*Unconstrained version*) foram selecionados vinte e sete (27) instâncias. Treze (13) instâncias para a versão de peças com valores não ponderados, GCUT1-GCUT13 tomados de (CINTRA et al., 2008) e catorze (14) instâncias selecionados para a versão de peças com valores ponderados, W1-W3 e UW1-UW11 tomados de (SONG et al., 2010). Estas 27 instâncias apresentam diferentes tipos de mochila com distribuições entre 10 e 70 tipos de peças, estes pertencem à categoria de problemas de corte de alta.

Trinta (30) instâncias foram selecionadas para os problemas com duas etapas de cortes. Quinze (15) instâncias para a versão de peças com valores ponderados, UW1-UW11 e UWL1-UWL4 e quinze (15) instâncias para a versão de peças com valores não ponderados UU1-UU11 e UUL1-UUL4. Estas 30 instâncias apresentam diferentes tipos de mochila com distribuições entre 25 e 200 peças, pertencentes à categoria de problemas de corte de alta complexidade matemática, estes casos de teste são tomados do banco de casos apresentado por (HIFI, 2001) e que está disponível em (HIFI, 1997b). Vinte e dois (22) destes são os mesmos usados para os problemas com três etapas de corte, tanto para as versões com valores ponderados e não ponderados (correspondentemente, UW1-UW11 e UU1-UU11). Além disto, para os problemas com padrões de corte tipo não guilhotina de primeira ordem e com um número de exemplares irrestrito, foram selecionados vinte e dois (22) instâncias, onze (11) instâncias para a versão de peças com valores ponderados (UW1-UW11) e onze (11)

instâncias para a versão de peças com valores não ponderados (UU1-UU11). Também apresentada por (HIFI, 2001) e disponíveis *on-line* em (HIFI, 1997b).

Por último, para os problemas com padrões de corte tipo não guilhotina de ordem superior e com um número de exemplares irrestrito, foram selecionados vinte casos (20) de teste, dez (10) instâncias para a versão de peças com valores não ponderados (ATP10 – ATP19) e dez (10) instâncias para a versão de peças com valores ponderados (ATP20 – ATP29), disponíveis em (ÁLVAREZ-VALDÉS et al., 2002). Para os problemas com um número de exemplares restrito, foram selecionados vinte e dois (22) instâncias, doze (12) instâncias para a versão de peças com valores ponderados (Beasley1 – Beasley12) disponíveis em (BEASLEY, 2004) e dez (10) instâncias para a versão de peças com valores não ponderados (Lai&Chan1-Lai&Chan3; Jakobs1 – Jakobs5; Leung1-Leung2) disponíveis em (LAI; CHAN, 1997; JAKOBS, 1996; LEUNG et al., 2003).

Existe uma grande quantidade de bancos de instâncias para os problemas apresentados neste estudo, porém após analisados muitos caem na categoria de problemas de empacotamento perfeito, ou seja, os casos teste foram desenvolvidos a partir da solução ótima do problema.

Adicionalmente, os bancos de instâncias utilizadas neste estudo não necessariamente apresentam um ótimo global nos limites inferiores propostos na literatura, por causa disto as metodologias exatas renunciam seu processo de otimização devido ao custo computacional.

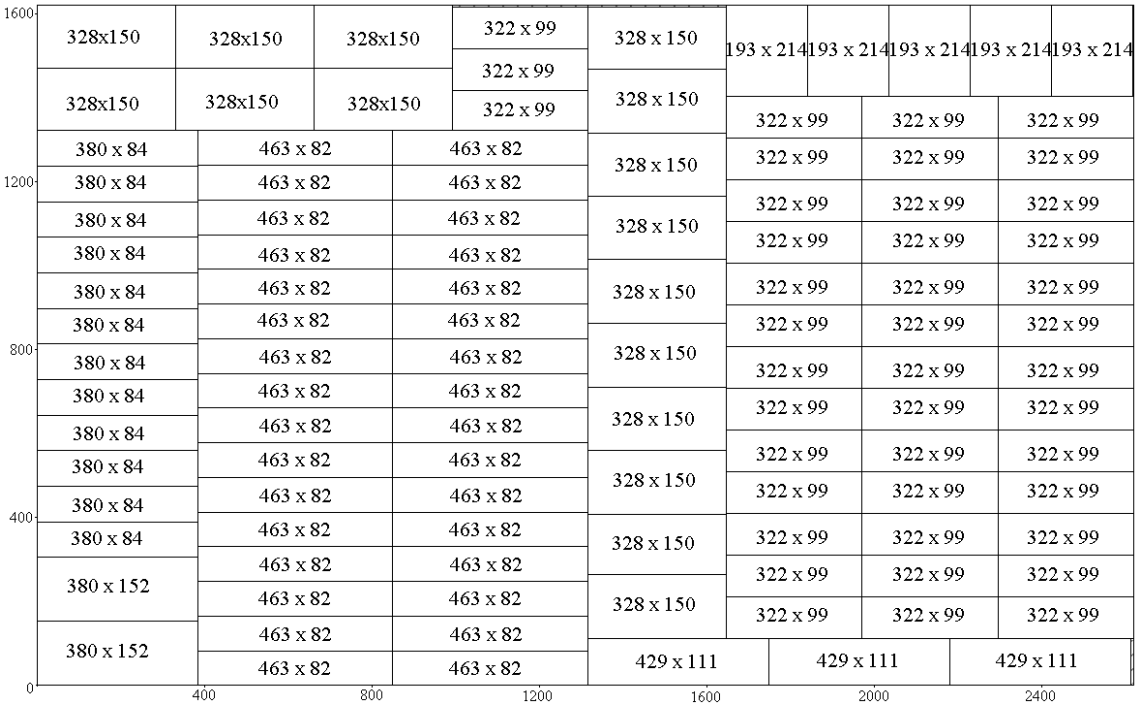
Nas Tabelas 15 - 47 apresentadas no Anexo foram selecionadas os valores objetivos das melhores soluções publicadas (MSP) na literatura. Nem sempre um algoritmo consegue obter as melhores soluções para todas as instancias, na maioria destas se encontra um empate, ou seja, diferentes algoritmos têm chegado à mesma solução. Neste trabalho se dá o crédito ao primeiro autor em relatar a melhor resposta no menor tempo computacional. Dessa forma a coluna MSP (Melhor Solução Publicada) sempre permite comparar-nos com a melhor solução publicada na literatura para cada caso de teste.

O algoritmos foi desenvolvido em C++ ® e executado em um computador com as especificações de um processador Pentium Core i7® de 2,99 GHz por núcleo de processamento e uma memória RAM de 4 GB.

Na Tabela 3 mostra-se o resumo dos resultados obtidos pela metodologia de solução proposta. Devido que não se tem um certificado de otimalidade de todos os casos apresentados na

literatura, neste estudo se mede o desempenho através do número de soluções ótimas alcançadas e o tempo médio computacional tomado pelo algoritmo de otimização. Baseados nestes critérios de desempenho, o algoritmo proposto aqui apresenta um bom comportamento nos problemas irrestritos (*unconstrained*) tipo guilhotina, e um comportamento regular em todas as versões tipo não guilhotina. Além disto, o algoritmo aqui apresenta um desempenho destacado no problema restrito (*constrained*) com restrições de padrão de corte de duas etapas. Por último, sobre os tempos computacionais não se pode fazer uma afirmação devido à diferença entre as arquiteturas de computação e as linguagens de programação, dado que neste estudo se optou por um único valor de parâmetros os tempos computacionais são quase indiferentes da versão do problema e realmente depende do tamanho da instância do problema e a relação intrínseca que guardam as dimensões das peças e a dimensão da chapa.

Figura 60 - Solução proposta para o caso ATP11 que pertence ao problema restrito, de peças com valor não ponderado. O valor objetivo apresentando neste trabalho é de 4.188.915



Fonte: Elaboração do próprio autor

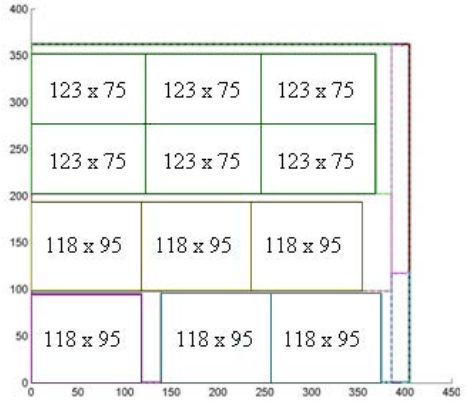
Algumas soluções são ilustradas nas Figuras 60 e 61. Por questões de dimensão os resultados restantes não são ilustrados neste capítulo e se encontram disponíveis no sítio web do grupo de pesquisa:

http://www.dee.feis.unesp.br/lapsee/arquivos/sistemastestes/knapsack_problem2D.html

Tabela 3 - Resultados alcançados pela metodologia proposta para os problemas da mochila bidimensional

			Unconstrained				Constrained			
			Weighted		Unweighted		Weighted		Unweighted	
			Fixed	Rotated	Fixed	Rotated	Fixed	Rotated	Fixed	Rotated
Guilhotina	Non-Staged	Tempo médio (s)	42.28		29.46	53.69	104.27		229.90	
		Média de tipos de peças	38.35		41	41	38.93		40.81	
		Número de Melhores Soluções Alcançadas	**14**		**13**	**13**	**10**		**6**	
		Número total de casos	14		13	13	15		11	
	2-Staged	Tempo médio (s)		102.86		111.86	23.38		42.78	72.16
		Média de tipos de peças		67.13		67.13			66.73	66.73
		Número de Melhores Soluções Alcançadas		**15**		**15**	**13**		**50**	**50**
		Número total de casos		15		15	13		50	50
	3-Staged	Tempo médio (s)	54.13	110.4	55.73	127.73	66.4		112.85	
		Média de tipos de peças	68.1	68.1	68.1	68.1	66.13		67.13	
		Número de Melhores Soluções Alcançadas	**15**	**15**	**15**	**15**	**15**		**19**	
		Número total de casos	15	15	15	15	20		20	
Não Guilhotina	First Order	Tempo médio (s)	134.09		134.27					
		Média de tipos de peças	27.84		27.84					
		Número de Melhores Soluções Alcançadas	**8**		**7**					
		Número total de casos	11		11					
	Superior Order	Tempo médio (s)	226		151.9		140.16		116.5	
		Média de tipos de peças	46.7		48.7		14.41		26.5	
		Número de Melhores Soluções Alcançadas	**5**		**8**		**6**		**1**	
		Número total de casos	10		10		12		10	

Figura 61 - Solução proposta para o caso CW9 que pertence ao problema tipo guilhotina restrita, com peças de valor ponderado. O valor objetivo obtido neste trabalho é de 10.748



Fonte: Elaboração do próprio autor

3.6 PROBLEMA DA EMBALAGEM

O problema da embalagem (mais conhecido como problema do *Bin-packing*) consiste em minimizar o material desperdiçado. A solução deste é de grande interesse no setor: industrial, comercial e acadêmico. Indicando alguns trabalhos desta temática com grande aplicação na indústria estão, os problemas: de desenho de chapas de metal (WY; KIM, 2010); de corte na indústria da lona para a confecção de barracas; toldos para Jipes (*Jeeps*) e outros (FARLEY, 1990); de corte na indústria do vidro (DYSON; GREGORY, 1976) e (MADSEN, 1979); de corte na indústria de roupas (FARLEY, 1988); o problema da perda residual no corte de papel corrugado (HAESSLER; TALBOT, 1990); de corte na indústria de madeira (MORABITO; GARCIA, 1997) e (VENKATESWARLU; MARTYN, 1992); de corte na indústria do papel (WESTERNLUND et al., 1995).

3.6.1 Introdução ao problema da embalagem

Nesta seção se analisa o caso específico onde se deve atender completamente a demanda de peças retangulares para ser localizadas em chapas retangulares com tamanho idênticos, cujo objetivo é minimizar o número de chapas necessárias para realizar a embalagem. Levando em conta as seguintes considerações:

i) Todas as chapas têm as mesmas dimensões, uma largura *W* e uma altura *H*.

ii) A orientação das peças pedidas, ou seja, a orientação da peça é fixa se: uma peça de altura *h* e largura *w* é diferente de uma peça de altura *w* e largura *h* (*fixed version*). Se

considerarmos que as dimensões (*h, w*) e (*w, h*) representam as dimensões da mesma peça, esta sendo tratado um problema com orientação variável das peças (*rotated version*).

iii) Os padrões de corte são do tipo guilhotina. Nestes cada corte produz dois sub-retângulos e vão de um extremo ao outro, do retângulo original.

Os problemas da embalagem apresentam uma alta complexidade matemática, estes são considerados *NP-Hard* num sentido forte. O problema da embalagem de uma dimensão foi provado *NP-Hard* por Garey e Johnson (1979) e Martello et al. (2003).

Que um problema este identificado nesta categoria não significa que não pode resolver-se, simplesmente quer dizer que se devem propor algoritmos de solução que explorem de forma eficiente a estrutura matemática do problema para que se encontrem soluções à maioria das instancias do problema em tempos de execução relativamente razoáveis.

Este problema foi amplamente estudado em diferentes campos da otimização, tais como: a otimização exata e a aproximada (heurísticas e meta-heurísticas). Lodi et al. (1999) e Lodi et al (2002) realizarem um excelente resumo do estado da arte do problema do *Bin-packing*, descrevem os limites disponíveis, algoritmos exatos e aproximados. Dos métodos exatos com melhor desempenho esta o proposto por Dell'Amico et al. (2002), este determina um limite inferior do problema com rotação de peças e o resolve com um algoritmo *Branch and bound*. Puchinger e Raidl (2006) consideram um problema típico de manufatura de vidro: *three-staged two-dimensional bin packing problem* aonde o número de cortes guilhotina não pode exceder três etapas de corte. Para resolver o problema estes desenharem dois modelos lineais inteiros mistos de tamanho polinomial e um algoritmo *Branch and price* baseado numa formulação de cobertura para o problema bidimensional.

Dentro dos algoritmos heurísticos com melhor desempenho estão: *Finite Best Fit Strip* (FBS) e *Finite First Fit* (FFF) apresentados por Berkey e Wang (1987) para o problema com orientação fixa das peças. Lodi et al. (1999) adaptam os algoritmos FBS e FFF para o problema com rotação das peças e apresentam uma aproximação *Floor Ceiling* (FC). Lodi et al. (1999) provam que o FC tem um melhor desempenho que o FBS e o FFF.

Existe uma grande quantidade de algoritmos aproximados que começaram a ser adaptados usando técnicas meta-heurísticas. Lodi et al. (2002) introduzem um algoritmo de busca no tabu, o qual explora o espaço de solução independente do problema utilizando um tamanho e uma estrutura de vizinhança dinâmica. A busca considera conceptos de heurísticas construtivas. Lodi et al. (2002) adaptam uma heurística construtiva para o problema com rotação das peças introduzindo o concepto de pseudo-peça e favorece a orientação vertical da peça quando é possível.

Faroe et al. (2003) estende a aproximação de Dowsland (1993) para o problema com orientação fixa, utilizando uma busca local guiada onde a vizinhança é explorada através de intercâmbios aleatórios de uma chapa a outra das peças empacotadas. A inviabilidade das peças sobrepostas é penalizada na função objetivo. A busca finaliza quando a incumbente trapaceia um valor proposto ou um período de tempo fixo é alcançado.

Novos algoritmos aproximados foram desenhados para resolver outras variantes do problema do *Bin-packing*. Por exemplo, Binkley e Hagiwara (2007) descrevem para o problema não guilhotina, a heurística de quatro esquinas que é usada em conjunto com um algoritmo paralelo autoadaptativo do algoritmo do arrefecimento simulado e do algoritmo genético.

As técnicas meta-heurísticas demonstram seu grande potencial como ferramentas de solução e tem variados campos de aplicação por sua eficiência em quanto a tempos de solução e qualidade das respostas obtidas. Neste trabalho se propõe um algoritmo baseado na técnica de otimização por enxame de partículas, para a resolução do problema da embalagem com restrições do padrão de corte tipo guilhotina e restrições de orientação das peças (fixa ou variável), além disto, será apresentada uma análise e uma comparação das respostas obtidas em relação às do estado da arte da literatura especializada. A estrutura deste capítulo é a seguinte: descrição do problema, modelo matemático representativo, metodologia de resolução, análise dos resultados e conclusões.

3.6.2 Descrição do problema do problema da embalagem

O problema da embalagem estudado neste trabalho se define como: corta um conjunto de retângulos denominados como chapas (mochilas, laminas ou objetos) de altura H e largura W, um conjunto de retângulos de cardinalidade n que se denominam peças (itens) de altura h_i e largura w_i, (onde $i = 1,..., n$). Onde o objetivo é determinado pela equação 7 e consiste em

minimizar o número de chapas necessárias para localizar e cortar a totalidade das peças demandadas, Z_l é uma variável binária que indica que chapa *l* foi ou não utilizada.

$$\min \quad \sum_{l=1}^{M} Z_l \tag{7}$$

Sujeito a:

10 – As peças empacotadas não devem ultrapassar os limites de cada chapa.

11 – As peças não devem se sobrepor entre si.

12 – Todas as peças devem ser atribuídas.

A Equação 7 e as expressões 10 –12 representam o problema geral, neste trabalho se estudam diferentes variantes que são encontradas na indústria, as características destas variantes podem ser incorporadas ao adicionar expressões ou modificar alguma delas.

As características deste problema são:

v) A orientação das peças: se a orientação das peças é fixa, não será necessário adicionar nenhuma expressão. Por outro lado, se é permitido a rotação das peças (90 graus), se deve realizar um pré-processo que identifique as peças *i* e *k* tais que $h_i = w_k$ e $w_i = h_k$, e fazer $b_i = b_i + b_k$, (ou seja, agrupar por tipos de peça) e adicionar a Expressão 13.

13 - As peças podem rodar 90°

vi) Os padrões de corte: Neste trabalho se limita ao uso dos padrões tipo guilhotina, para isto é necessário adicionar a Expressão 14.

14 - Unicamente cortes tipo guilhotina são permitidos.

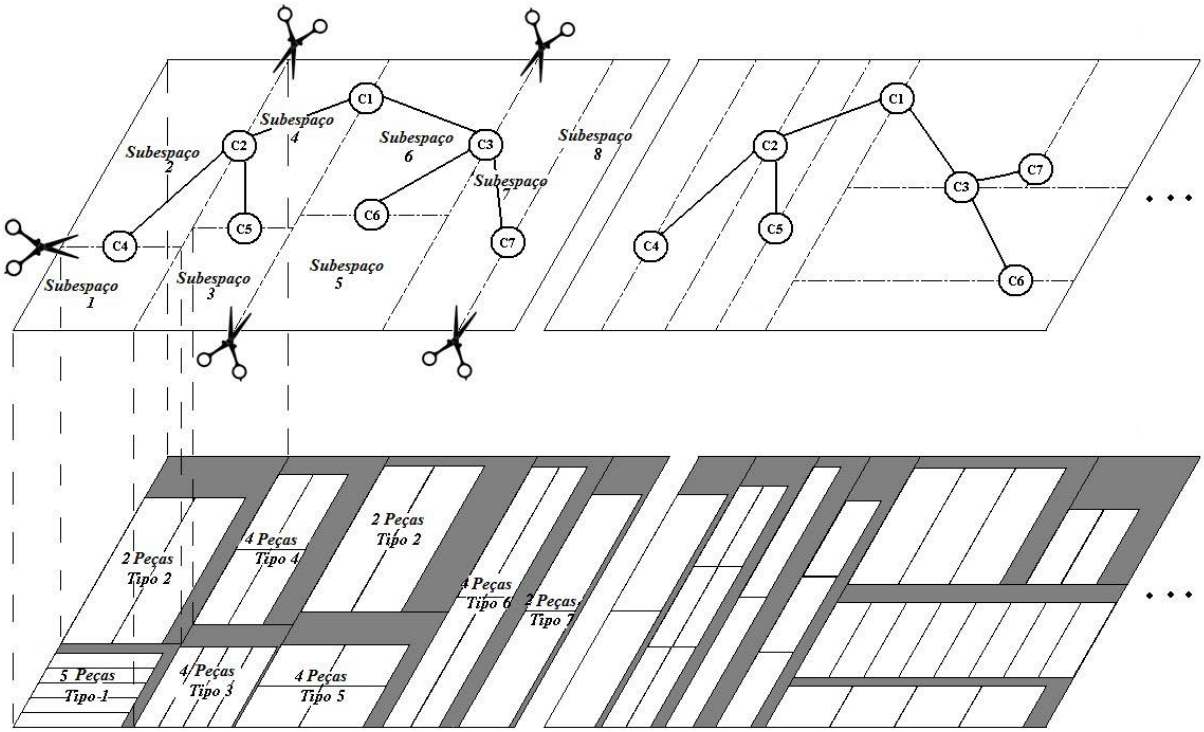
Isto significa que o tipo de padrão de corte é uma restrição forte neste trabalho.

3.6.3 Metodologia de resolução do problema da embalagem

Nesta seção se propõe uma metodologia de resolução baseada no problema da mochila apresentada na Seção 3.4. Por tanto, não é necessário indicar novamente: o esquema de otimização, a codificação, a representação do espaço livre dos espaços máximos residuais, o algoritmo heurístico construtivo de criação de capas e o cálculo da função objetivo. Isto implica que serão usados da mesma forma que foi proposto na Seção 3.4.

Neste trabalho o problema da embalagem se resolve como a solução de *l* problemas da mochila bidimensional consecutivos, sendo *l* o número mínimo de chapas necessárias para empacotar a totalidade das peças (ver Figura 62). Portanto, o esquema de otimização consiste em executar o algoritmo $A_{PSO+VND+TF}$ chapa por chapa e ir atualizando o inventário das peças utilizadas em cada chapa, desta forma a demanda das peças diminuirá durante a evolução do processo de otimização.

Figura 62 - A árvore de cortes e localização das peças chapa por chapa



Fonte: Elaboração do próprio autor

3.6.4 Análise dos resultados do problema da embalagem

Para validar a qualidade das respostas obtidas do problema solucionado, foi selecionado tudo o banco de instâncias (500 instâncias) proposto por Berkey e Wang (1987) (classes 1–6) e Lodi et al. (1997) (classes 7–10), ao mesmo tempo foram atualizadas as publicações mais notórias na literatura. As 500 instâncias tomadas do banco de instâncias por Berkey e Wang (1987) e Lodi et al. (1997), estão disponíveis *on-line* em (LODI et al., 2002). Diferentes estudos têm utilizado estas instâncias de teste para realizar suas análises em uma espécie de estudo de *benchmark* das metodologias propostas. Uma descrição da criação dos casos de teste é apresentada em Lodi et al. (1997). O algoritmo foi desenvolvido em Delphi 7.0 ® e executado sobre um computador com um processador Pentium IV® de 3,0 GHz e uma memória RAM de 1 GB.

Tabela 4 - Comparação com as metodologias mais notáveis da literatura. *Fixed version.*

Classe	**Tamanho (L x W)**	**n**	**CHBP-CF**	**CFIH+J4**	**$A_{PSO+VND+TF}$**
1	10x10	20	71	71	71
		40	134	135	**135**
		60	201	201	201
		80	275	275	**282**
		100	321	322	**333**
2	30x30	20	10	10	**12**
		40	20	20	20
		60	26	27	**27**
		80	33	32	**33**
		100	39	40	**43**
3	40x40	20	52	53	**55**
		40	97	96	**97**
		60	140	141	140
		80	196	195	**198**
		100	230	226	**236**
4	100x100	20	10	10	10
		40	19	19	19
		60	25	25	**26**
		80	33	33	33
		100	39	39	**40**
5	100x100	20	65	66	**73**
		40	121	120	**130**
		60	183	182	182
		80	247	248	**251**
		100	288	290	**304**
6	300x300	20	10	10	10
		40	19	19	19
		60	22	22	22
		80	30	30	30
		100	34	34	34
7	100x100	20	55	56	55
		40	112	115	**117**
		60	160	161	**162**
		80	233	232	232
		100	275	274	**280**
8	100x100	20	58	60	**65**
		40	114	116	114
		60	163	165	163
		80	226	227	226
		100	279	281	**285**
9	100x100	20	143	143	143
		40	279	278	278
		60	438	437	**441**
		80	577	577	577
		100	695	695	**702**
10	100x100	20	44	43	43
		40	74	75	**81**
		60	103	104	**104**
		80	130	132	130
		100	163	163	**166**
Total			7311	7325	7430

Tabela 5 - Comparação com as metodologias mais notáveis da literatura. *Rotated version*

Classe	**Tamanho (L x W)**	**n**	**CHBP-CF**	**CFIH+J4**	**$A_{PSO+VND+TF}$**
1	10x10	20	66	66	**71**
		40	129	129	**134**
		60	195	195	**201**
		80	271	271	**275**
		100	314	314	**321**
2	30x30	20	10	10	10
		40	19	19	**20**
		60	25	25	**26**
		80	31	32	**32**
		100	39	39	39
3	40x40	20	48	48	**52**
		40	94	94	**96**
		60	136	136	**140**
		80	186	185	**195**
		100	223	223	**226**
4	100x100	20	10	10	10
		40	19	19	19
		60	25	25	25
		80	33	33	33
		100	38	38	**39**
5	100x100	20	59	59	**65**
		40	115	117	**119**
		60	176	175	**182**
		80	240	241	**247**
		100	282	281	**288**
6	300x300	20	10	10	10
		40	18	18	**19**
		60	21	21	**22**
		80	30	30	30
		100	34	34	34
7	100x100	20	52	52	**55**
		40	104	106	**112**
		60	148	151	**160**
		80	211	214	**232**
		100	255	258	**274**
8	100x100	20	53	53	**58**
		40	105	105	**114**
		60	150	152	**163**
		80	210	211	**226**
		100	258	258	**279**
9	100x100	20	143	143	143
		40	275	275	**278**
		60	435	435	**437**
		80	573	573	**577**
		100	693	693	**695**
10	100x100	20	41	42	**43**
		40	73	73	**74**
		60	100	101	**103**
		80	130	129	**130**
		100	159	159	**164**
Total			7064	7080	7297

Nas tabelas 4 e 5 mostra diferentes algoritmos e seus resultados para cada grupo de instâncias. Para as 500 instâncias de Lodi et al. (2002) os resultados estão determinados pelo número de chapas utilizadas para cada grupo de 10 instâncias. Por colunas apresentam-se: o algoritmo heurístico (CHBP) proposto por Charalambous e Fleszar (2011), o algoritmo heurístico (CFIH+J4) sugerido por Fleszar (2013), e afinal, o algoritmo híbrido de PSO, VND e o fator de turbulência ($A_{PSO+VND+TF}$) proposto neste trabalho. Deve-se notar que existem outros trabalhos na literatura, embora não são apresentados aqui por sua forma de publicar os resultados, já que ao usar diferentes limites inferiores presta-se para erros na transcrição dos resultados, por este motivo, porém sem perder a qualidade da comparação aqui realizada, se utiliza o número de chapas como unidade para comparar resultados.

Na Tabela 6 são resumidas e comparadas as médias dos tempos computacionais requeridos pelo algoritmo para cada problema. Nesta é evidente que o algoritmo aqui proposto é inferior em tempos de resposta.

Tabela 6 - Comparação dos tempos de computo do algoritmo proposto. Unidades em segundos

Problema	**CHBP-CF**	**CFIH+J4**	**$A_{PSO+VND+TF}$**
Fixed version	33	<1	210
Rotated version	66	<1	299

Dado que em aplicações reais consideram-se no processo de corte o empacotamento os limites máximos de desperdício de 30% do material. Pode-se concluir que aplicando esta metodologia se obtém uma redução das perdas do material de mais de 50%. O inconveniente na aplicação desta metodologia é o alto tempo de computacional, quando se deseja resolver um problema em tempo real.

3.7 CONCLUSÕES

Neste Capítulo desenvolveu-se uma metodologia de solução para os problemas da mochila bidimensional, que utiliza uma codificação de árvore de cortes e usa a representação dos espaços máximos para administrar os espaços residuais, enquanto o processo geral é administrado mediante um algoritmo híbrido de otimização por enxame de partículas, busca em vizinhança variável e algoritmos genéticos.

Desenvolve-se uma metodologia de resolução para os problemas da embalagem baseada na metodologia proposta para o problema da mochila bidimensional. Esta metodologia consegue

ser flexível e robusta para solucionar uma grande variedade de variantes do problema propostos neste trabalho, além de alcançar resultados satisfatórios.

Foi utilizado um tipo de codificação em árvores, chamada de árvores de cortes, a qual se encarrega de subdividir o problema original em pequenos subproblemas da mesma espécie. Isto é conseguido porque nos nós pais da árvore se representam as distâncias e as orientações dos cortes que se devem realizar sobre a chapa, enquanto nos nós folha armazenam-se as dimensões das chapas dos subproblemas resultantes. Esta proposta de codificação apresenta um bom desempenho para este tipo de problemas, pois consegue reduzir o espaço de busca com um risco baixo de perder soluções de boa qualidade. Este esquema de recursividade (subdividir o problema em pequenos subproblemas) apresenta uma grande flexibilidade, isto significa que pode ser usada em metodologias que utilizem outras técnicas de otimização tais como: *Branch and bound*, Búsqueda Tabú, GRASP, Colônia de formigas, Algoritmos Genéticos, etc.

Foi desenvolvido um algoritmo de otimização que combina as principais características dos algoritmos de otimização por enxame de partículas, busca em vizinhança variável e algoritmos genéticos. O primeiro se considera como o algoritmo principal, o segundo desempenha o papel de limitador das características da solução que se devem atualizar em cada iteração (isto faz que aumente o processo de exploração) e o terceiro, é utilizado como mecanismo de perturbação especializado para realizar buscas locais (efetuando mudanças aleatório-controladas nas soluções através do operador de mutação dos genéticos) que representam o concepto de fator de turbulência do voo original do algoritmo PSO.

Os tempos computacionais requeridos pela metodologia, em algumas vezes, são menores aos publicados na literatura especializada, porém devido às diferenças entre as arquiteturas de computo e as linguagens de programação não se pode concluir com base neste índice. Apesar disto, os tempos computacionais utilizados pela metodologia proposta são razoáveis.

Estes problemas são aplicáveis em diversos setores da economia, entre os que se destacam os setores da indústria de transporte e comércio. Assim por exemplo, sua aplicação se apresenta na indústria têxtil, metal mecânica, papeleira, vidreira, transporte e armazenagem de mercadorias, entre outras.

A metodologia proposta é aplicável em problemas de: *3D Bin-packing*, *Strip packing* e entre outros. Também pode extendida aos problemas de carregamento de *pallets*

4 PROBLEMA DE CARREGAMENTO DO CONTÊINER

O problema de carregamento de contêineres é um problema clássico na pesquisa operacional e de grande importância na indústria, a pesar disto não é comum que a comunidade acadêmica incorpore nos estudos todas as características importantes que representam situações práticas destes problemas, como orientação das caixas, limites de resistência das caixas ao empilhamento, limite de peso do carregamento suportado pelo contêiner, estabilidade do carregamento e carregamento com múltiplos destinos.

Este problema comummente é denominado como Problema de carregamento do contêiner com restrições de *multi-drop*. O problema consiste em maximizar o volume total ocupado pelo carregamento empacotado, de forma tal que, se cumpram as restrições de empacotamento e a restrição de *multi-drop* que impõe que, as caixas que alcançam seu destino final devem ser descarregadas desorganizar a carga que ainda deve permanecer.

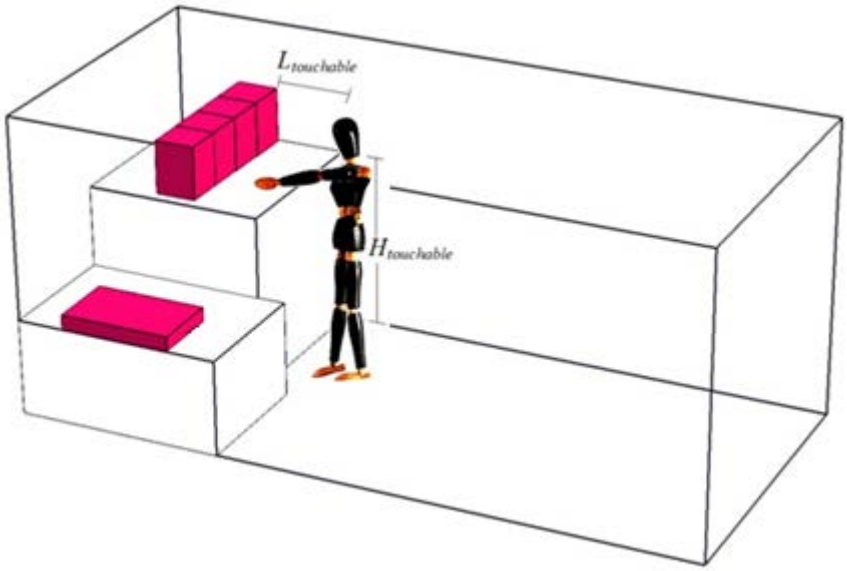
Para isto, se utiliza a representação dos espaços-máximos e um algoritmo construtivo aleatório, que permite construir soluções que cumprem em sua totalidade com as restrições impostas. O algoritmo está baseado na meta-heurística GRASP (do inglês, *Greedy Random Adaptive Search Procedure*) e incluí um conjunto de movimentos de melhoramento, os quais estão direitamente relacionados às restrições do problema apresentadas neste trabalho, tais como construir pilhas, blocos ou filas e compactar soluções por cliente.

Por último, é apresentado e discutido um estudo computacional, utilizando as distintas bases de instâncias reais e de teste apresentadas na literatura especializada.

4.1 INTRODUÇÃO

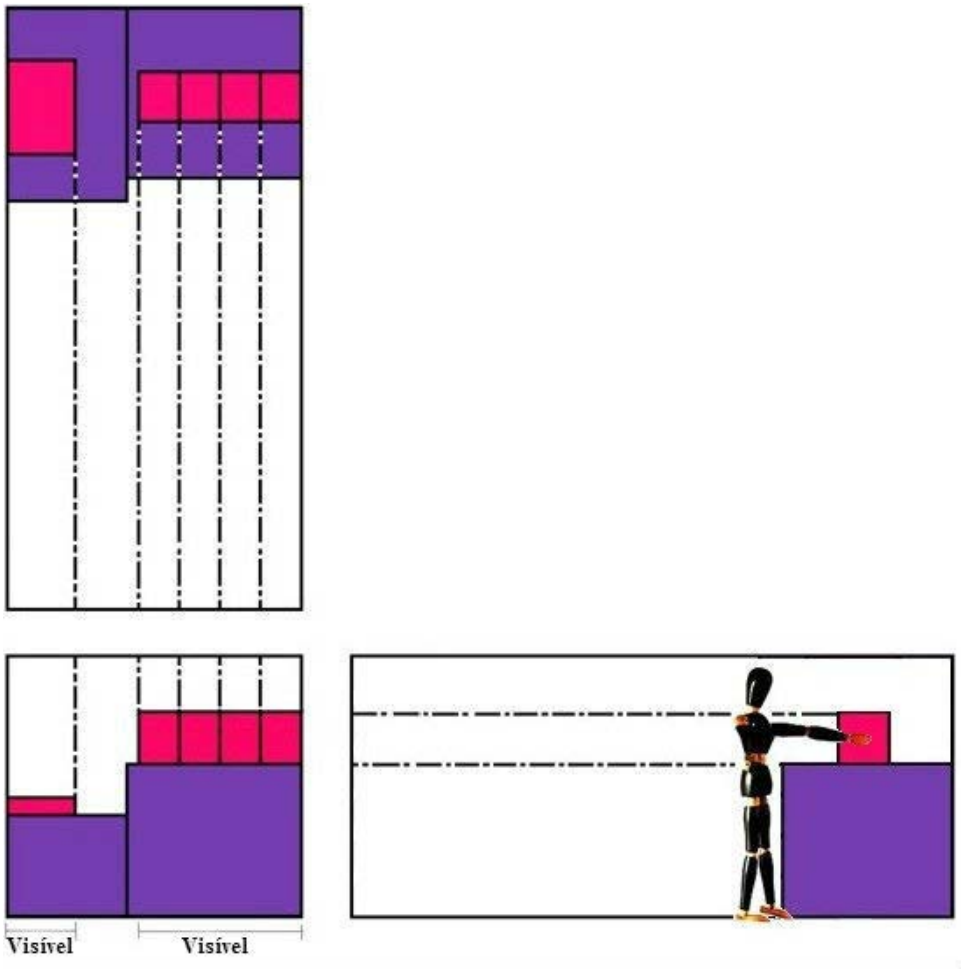
Neste trabalho, consideramos todas as restrições anteriormente mencionadas. Formalmente, este problema é conhecido como o problema de localização tridimensional (*Single Large Object Placement Problem*, SLOPP) ou o problema da mochila tridimensional (*Single Knapsack Problem,* SKP) com múltiplos destinos (*multi-drop*). O objetivo do problema é maximizar o valor do carregamento empacotado, porém garantindo satisfazer todo o conjunto de restrições.

Figura 63 - *Multi-drop* para Liu et al. (2011)



Fonte: Elaboração do próprio autor

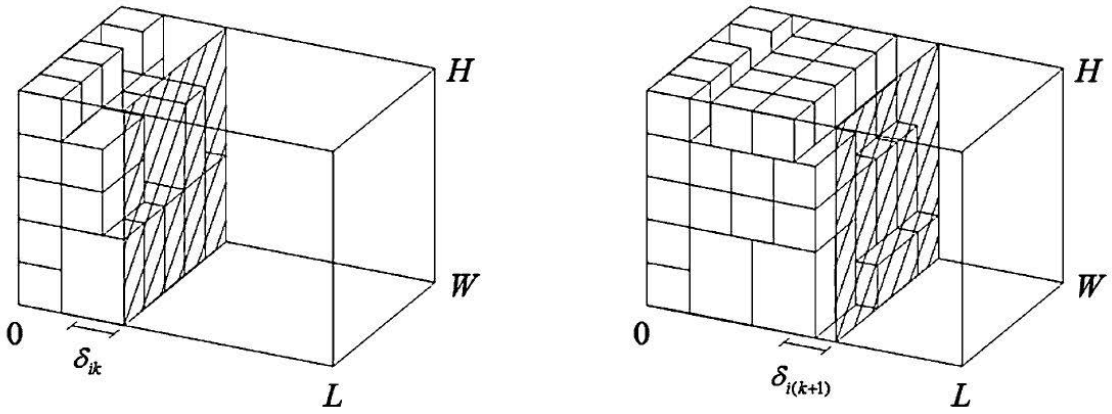
Figura 64 - Vistas: superior, frontal, e lateral direita, da Figura 63



Fonte: Elaboração do próprio autor

Especificamente, a restrição de múltiplos destinos se refere ao fato de ter sido estabelecido para o contêiner (ou caminhão) um roteiro de visita para o conjunto de destinos (clientes), isto especifica a ordem de descarregamento das caixas empacotadas, restringindo assim que o conjunto de caixas que vai para diferentes clientes devam ir localizados de forma tal que, permita a descarregamento das caixas de um cliente sem ter que mover as caixas doutros clientes.

Figura 65 - *Multi-drop* para Junqueira et al., (2012a)

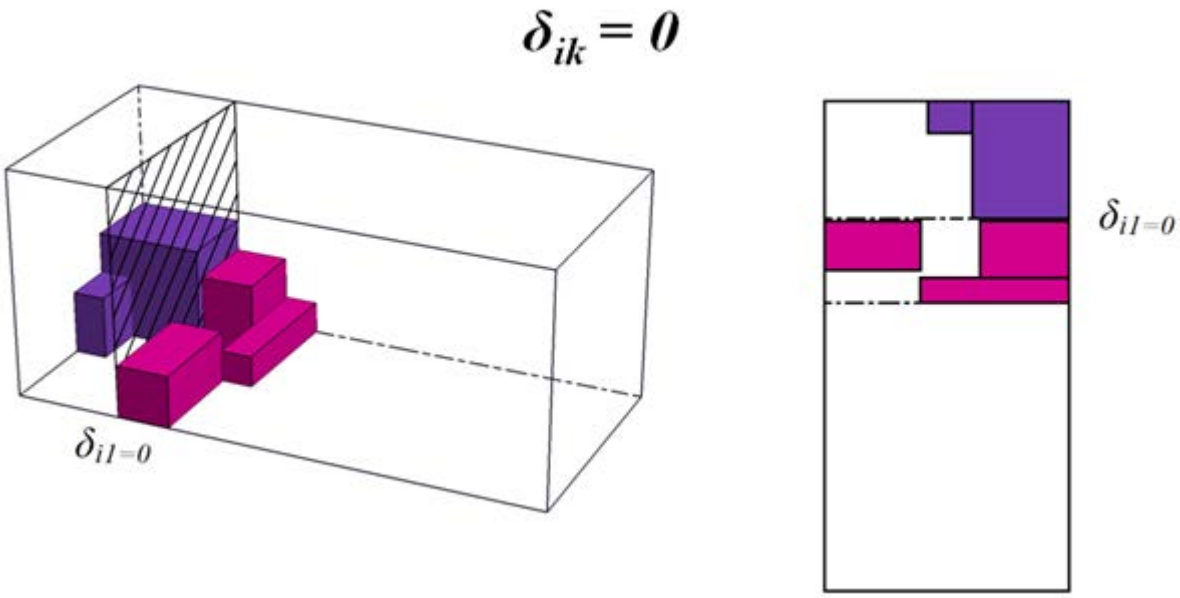


Fonte: Elaboração do próprio autor

Nos trabalhos anteriores que consideram a restrição de *multi-drop* se apresenta uma discussão devido a que cada um deles define e especifica o que representa uma situação *multi-drop*. Entre os artigos que se ressaltam estão Liu et al., (2011) que definem um empacotamento *multi-drop* impondo restrições de localização do carregamento baseadas nas definições de invisibilidade e inatingibilidade das caixas empacotadas (ver Figura 63), onde uma caixa é invisível se a totalidade da face frontal dianteira desta, não se pode ver da porta do contêiner e é inalcançável se está localizada fora dos limites de alcance do sistema de descarregamento (sejam os braços e altura de um trabalhador ou uma máquina empilhadeira, ver Figura 64).

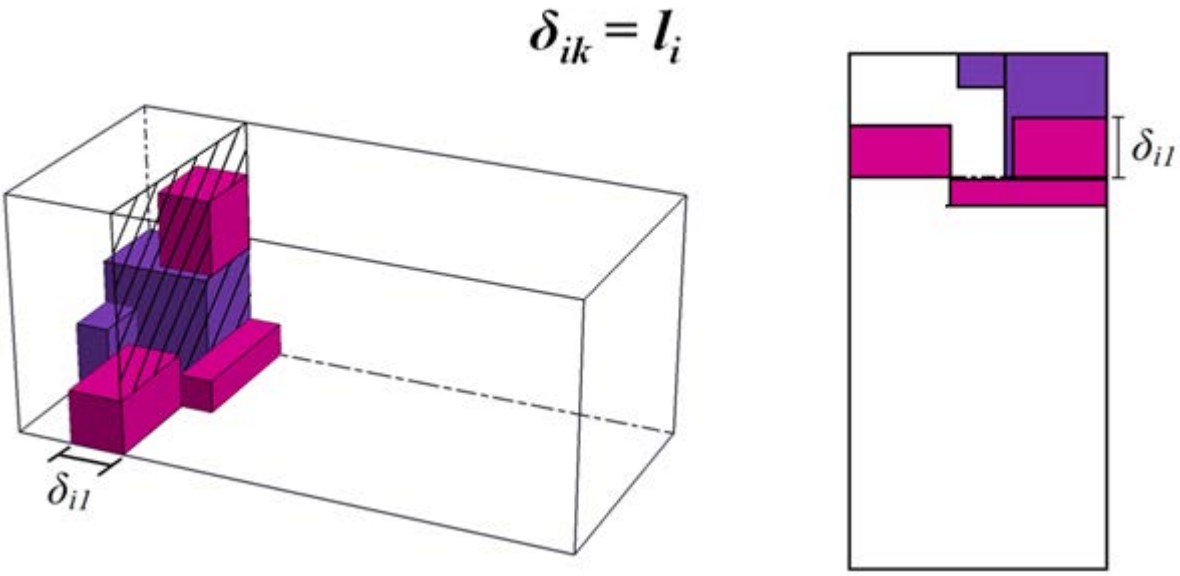
Junqueira, et al. (2012a), definem um empacotamento *multi-drop* impondo restrições de localização ao introduzir uma fronteira imaginária (ou plano de separação ao longo do contêiner) que separa os carregamentos empacotados de cada cliente, esta se estabelece a partir da caixa mais próxima à porta do contêiner, gerando assim uma espécie de cortina ou parede virtual (ver Figura 65). Depois, define um parâmetro δ que representa o comprimento do braço do homem encarregado ou o comprimento da forquilha da empilhadeira. Onde, δ_{ik} são as unidades de comprimento que caixas do cliente k podem ultrapassar a fronteira i (ver Figuras 66 e 67).

Figura 66 - Empacotamento *multi-drop* segundo Junqueira com $\delta = 0$



Fonte: Elaboração do próprio autor

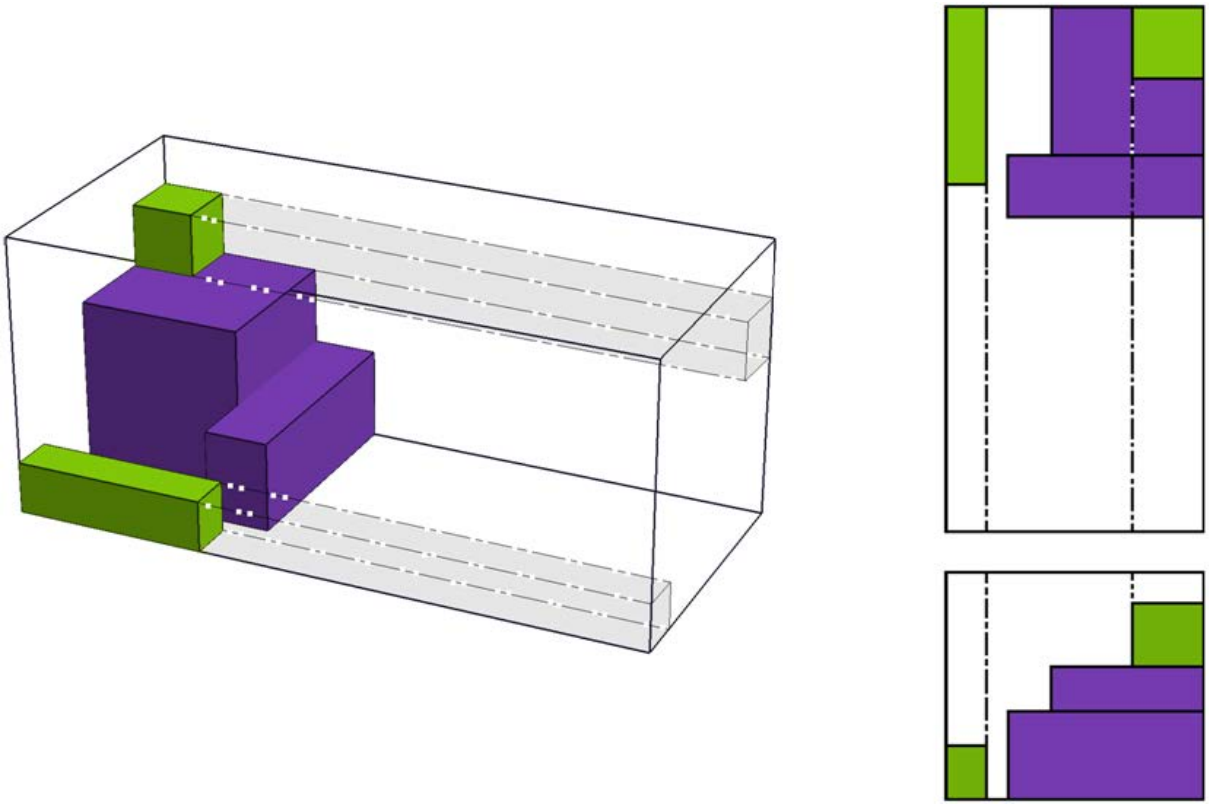
Figura 67 - Empacotamento *multi-drop* segundo Junqueira com $\delta = l_i$



Fonte: Elaboração do próprio autor

Christensen e Rousøe (2009) e Ceschia e Schaerf (2013), apresentam uma definição do que é um empacotamento *multi-drop* factível, o qual consiste que no momento de descarregar as caixas de um cliente acima destas não tenha nenhuma caixa que ainda não tem sido descarregada, além disto, deve existir um corredor livre entre as caixas e a porta do contêiner (ou seja, a face frontal dianteira das caixas é totalmente visível desde a porta, ver Figura 68).

Figura 68 - *Multi-drop* para Christensen e Rousøe (2009) e Ceschia e Schaerf (2013)



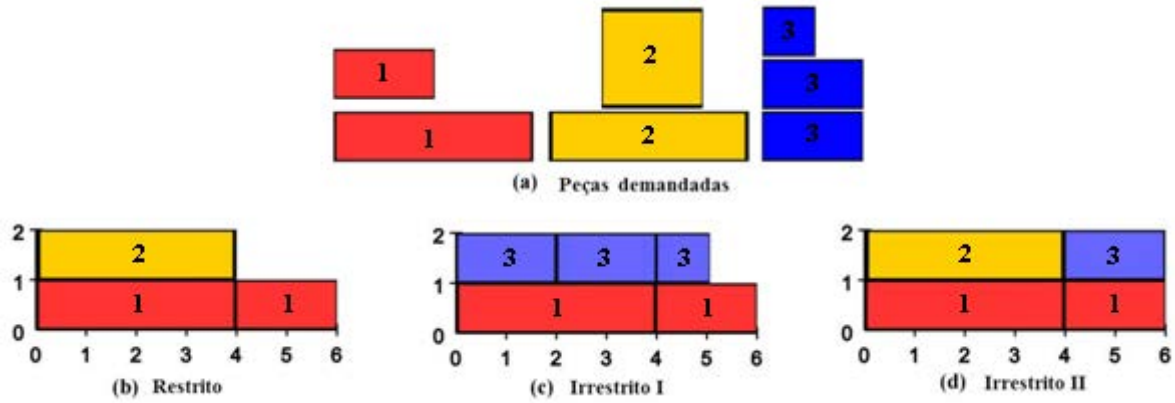
Fonte: Elaboração do próprio autor

As diferenças entre autores no tratamento das restrições de *multi-drop* dificulta a geração de uma metodologia de solução general. Pelo que neste trabalho, no momento de comparar com trabalhos anteriores é adaptado o algoritmo para que cumpra o mesmo cenário e possa realizar-se uma comparação justa.

Isto acontece também no tratamento das restrições de envio completo, onde este pode ser restrito ou irrestrito. Para isto o algoritmo proposto tem sido adaptado tanto à versão do problema resolvida pelo Christensen e Rousøe (2009) e Ceschia e Schaerf (2013) aonde os pedidos do cliente devem ser atendidos em sua totalidade (ver Figura 69b), tanto na versão tratada por Liu et al., (2011) e Junqueira et al., (2012a) onde não é necessário empacotar a totalidade da demanda de cada cliente, já que se considera que é possível enviar as caixas restantes em outro contêiner (ver Figura 69d). Devem notar também, que existe mais uma variante da versão irrestrita que se aplica no problema de roteamento de veículos com restrições de empacotamento tridimensional (3L-CVRP do inglês *three-dimensional loading capacitated vehicle routing problem*). Nesta versão não é permitido visitar um cliente mais de uma vez, isto implica que, embora existindo espaço livre no contêiner se a totalidade da demanda do cliente não se consegue empacotar a este cliente não poderá ser atendido (ver

Figura 69c.), como acontece em Gendreau et al., (2006), Tarantilis et al., (2009), Bortfeldt (2012) e Ceschia et al., (2013).

Figura 69 - Situações das restrições de empacotamento de envios completos



Fonte: Elaboração do próprio autor

Na Figura 69 se ilustra um exemplo dos cenários das restrições de empacotamento de envios completos. Na Figura 69a temos a demanda dos três clientes, e um contêiner bidimensional de dimensões (*6, 2*). Na Figura 69b se tem a solução ótima para a versão restrita alcançando um valor de função objetivo igual a 10 unidades de área. A Figura 69c é a solução ótima (da variante irrestrita I, U1) usando as caixas dos clientes 1 e 3, alcançando um valor objetivo igual a 11 unidades de área. Finalmente, na Figura 69d se ilustra uma das soluções ótimas da variante menos restrita (irrestrita II, U2) aonde se consegue alcançar um valor objetivo de 12 unidades (neste caso um empacotamento sem perdas), porém se termina utilizando parcialmente as demandas dos clientes, isto representa penalidades ou compensações monetárias para com os clientes.

4.2 DESCRIÇÃO DO PROBLEMA

O problema de carregamento do contêiner estudado neste trabalho se define como empacotar dentro de um paralelepípedo que se denomina contêiner com dimensões (*L*, *W* e *H*, comprimento, largura e altura, respectivamente), um conjunto de paralelepípedos de diferentes tipos (*i=1,…, n*) que se denominam caixas com dimensões (l_i, w_i, h_i), número de exemplares q_i, orientações de posicionamento o_{ij} (*j=1, 2,3*), um peso w_i, um limite de suporte de peso por face b_{ij} (j=1,) e um destino d_i (cliente ou sequenciador de descarregamento). Onde o valor da função objetivo é determinado pela Equação 8 e consiste em encontrar um empacotamento ortogonal das caixas de forma que se maximize o volume de utilização do

contêiner, z_i é uma variável inteira que indica o número de exemplares da caixa *i* que devem ser empacotados.

$$max \sum_{i=1}^{n} l_i \cdot w_i \cdot h_i \cdot z_i$$	*(8)*

Sujeito a:

15 - As caixas empacotadas não devem superar os limites do contêiner.

16 - As caixas não devem sobrepor-se entre elas.

17 - $z_i \leq q_i$, as caixas atribuídas não devem ultrapassar a demanda por tipos das peças.

As Expressões 15 – 17 representam o problema general, neste trabalho foram estudadas as diferentes restrições que representam as situações reais encontradas na indústria. As características deste problema são as seguintes:

i) Suporte completo: a base de cada caixa tem que estar localizada no piso do contêiner ou completamente na superfície doutro arranjo de caixas.

ii) Orientações de posicionamento: Cada caixa tem um conjunto de orientações permitidas, devido a seu conteúdo ou sua estrutura. As orientações são denotadas por o_{ij}, onde *i=1,…, n* representa o tipo de caixa e *j=1, 2, 3* a orientação, que definem a dimensão da caixa que pode ser posicionada para cima: $o_{i1} = 1$ se a dimensão *l* (comprimento) da caixa pode ser localizada em posição vertical e *0* (zero) se não pode erguer-se, $o_{i2} = 1$ se a dimensão *w* (largura) da caixa pode ser localizada em posição vertical e *0* (zero) se não se pode erguer, finalmente $o_{i3} = 1$ se a dimensão *h* (altura) da caixa pode ser localizada em posição vertical e *0* (zero) se não pode erguer-se. Ao menos um destes parâmetros deve ser *1* para cada tipo de caixa.

iii) Limites de empilhamento: Cada caixa localizada numa de suas orientações permitida pode suportar um peso máximo b_{ij} sobre sua superfície, expressado em gr/cm^2, onde *i=1,…, n* representa o tipo de caixa e *j=1, 2, 3* a orientação. Como Bischoff (2006) indica, existem diferentes formas nas quais o peso da caixa é transmitido para baixo sobre as caixas que a estão suportando. Na prática, o peso é transmitido dependendo da rigidez da superfície em contato: quanto mais rígido é, mais esparzido é o peso sobre toda a superfície. Quando são usados materiais macios como papelão, o peso da caixa na parte superior é transmitido para baixo somente na área de contato. Neste

trabalho se assume esta última situação, além disto, quando uma caixa *k* é colocada acima doutra caixa *i*, seu peso w_k será dividido pela área de sua base para calcular a pressão, p_k em gr/cm^2, exercida sobre as caixas que a estão suportando. Se esta pressão p_k excede o limite de empilhamento da caixa inferior, b_{ij}, o empacotamento é infactível. De outra maneira, o limite de empilhamento da caixa *i* será reduzido por p_k.

iv) *Multi-drop*: Se um contêiner leva caixas a diferentes destinos D, e se tem uma ordem (roteiro) de visita destes, as caixas deveriam ser carregadas de forma tal que, o empacotamento conserve a ordem de entrega estabelecida, evitando que caixas de um destino posterior devam ser mexidas para descarregar as caixas dos destinos anteriores (primeiros em sequência de descarregamento). Esta é uma definição informal porque existem diferentes alternativas de descarregar as caixas sem mexer as caixas dos clientes posteriores (como tem sido ilustrado na seção 4.1).

v) Envios completos: Quando num contêiner existem carregamentos de diferentes clientes *D*, pode ser exigido o embarque completo do pedido de cada cliente. Isto combinado com a restrição de *multi-drop* implica que, na sequencia de empacotamento não seria possível colocar uma caixa do cliente *j* até que todas as caixas do cliente *i* tenham sido empacotadas ($j > i$). Por outro lado, quando não é exigido um envio completo do pedido dos clientes, é possível empacotar elementos do cliente *j* ainda quando existam elementos do cliente *i* sem empacotar. Estes últimos (as caixas não entregues a um cliente) acabam sendo refletidos em multas ou compensações monetárias por parte da empresa que não consegue enviar todos os itens requeridos.

4.3 MODELO MATEMÁTICO

Devido à alta complexidade do problema de carregamento do contêiner e que pertence à família de problemas *NP-Hard*, existem poucos modelos de programação inteira mista para resolver o problema. Porém, deve-se notar que na literatura existem modelos pertencentes ao problema bidimensional de corte e empacotamento que podem ser adaptados ao CLP.

O modelo proposto por Chen et al. (1993) parece ser o mais representativo para a comunidade acadêmica para este problema, Chen et al., formula um modelo de programação inteiro mista para o problema de carregamento do contêiner e múltiplos contêineres, com o qual consegue resolver otimamente problemas de pequeno porte e demonstra assim, que o modelo apresenta

soluções factíveis do problema. A validação deste modelo é feita usando somente seis caixas e se tem imposto um limite de tempo de computacional de 900 segundos. Porém sua conclusão final acaba afirmando que é necessário um melhor procedimento de solução na hora de resolver instâncias de tamanho real do problema. Infelizmente, nenhuma das considerações reais abordadas neste trabalho é levada em conta no modelo de Chen et al. (1993), excetuando as restrições de orientação das caixas. Mas deve-se ressaltar que este modelo é extensível aos problemas onde a restrição de distribuição de peso é imposta.

Por outro lado, Junqueira et al. (2012a; 2012b e 2013) apresentam modelos de programação linear inteira mista onde são incluídas restrições como: a estabilidade vertical e horizontal do carregamento, os limites de empilhamento das caixas e carregamento com múltiplos destinos. Este modelo é uma adaptação do modelo de Beasley (1985), que é um modelo de programação linear inteira 0-1, formulado para resolver o problema de corte bidimensional. Os modelos formulados por Junqueira são testados usando poucas caixas e se resolvem impondo um limite de tempo computacional de 3600 segundos. Conseguindo alcançar para a maioria das instâncias de teste soluções ótimas e factíveis. Infelizmente, se concluí também que para resolver instâncias de tamanho real do problema é necessário melhorar o procedimento de resolução.

Deve-se notar também que existem outras transformações na literatura, a maioria inspiradas nos modelos matemáticos formulados para os problemas de corte e empacotamento bidimensional: Scheithauer (1998) apresenta duas relaxações do problema tridimensional: uma relaxação por barras e uma relaxação por camadas para resolver o CLP. Tsai et al. (1993), apresenta um modelo de programação linear inteira mista 0-1 para o problema de carregamento de *pallets*. Hadjiconstantinou e Christofides (1995) apresentam um modelo de programação linear inteira 0-1, para o problema da mochila bidimensional que acaba sendo adaptável ao CLP. Beasley (2004) apresenta um modelo de programação não linear inteira mista 0-1 para resolver o problema da mochila bidimensional que também pode ser transformado para descrever o CLP. Também é interessante, o modelo para empacotamento de polígonos convexos e não convexos apresentado pelo Scheithauer e Terno (1993), o modelo de Padberg (2000) que é um extensão do modelo de Fasano (1999) e o modelo proposto por Martins (2003) que consegue um bom desempenho para o problema de carregamento de *pallets*.

4.4 METODOLOGIA DE RESOLUÇÃO

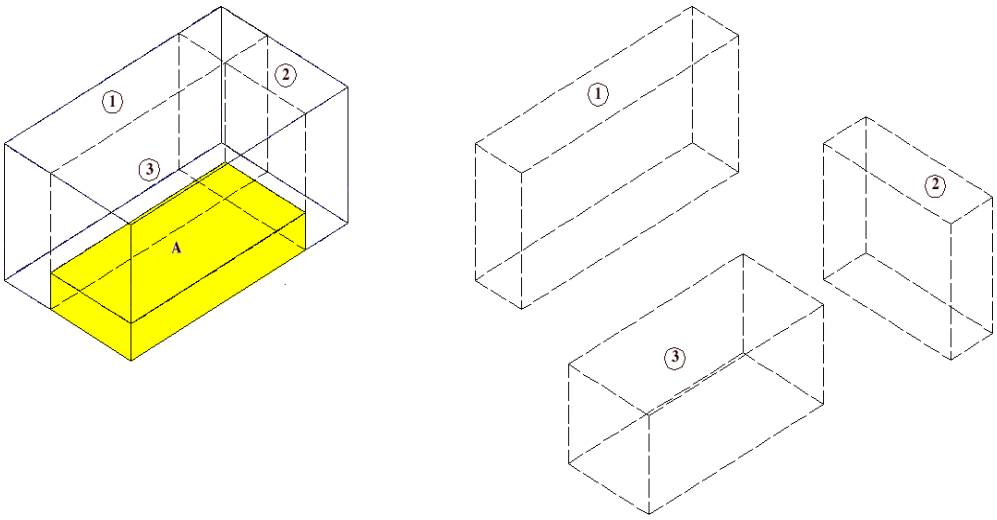
Neste trabalho se propõe um algoritmo GRASP (*Greedy Randomized Adaptive Search Procedure*) baseado na representação dos espaços-máximos, o qual na fase de construção permite obter soluções totalmente factíveis através do controle na geração e atualização dos espaços-máximos, satisfazendo assim as restrições de orientação das caixas, limite de empilhamento das caixas, limite de peso do carregamento, estabilidade do carregamento e padrões *multi-drop* (carregamento com múltiplos destinos). A estratégia de busca tipo GRASP, permite embaralhar a seleção do tipo e a quantidade de caixas a localizar em cada espaço-máximo. Por outra parte, na fase de busca local se realizam movimentos de melhora como a compressão de carregamentos por cliente, e o esvaziado e recheado determinista.

GRASP

O algoritmo GRASP foi desenvolvido por Feo e Resende (1989) para resolver problemas de otimização combinatória difíceis. Diferentes estudos demonstram sua qualidade e robustez, para uma introdução e atualização o leitor é convidado a ver (RESENDE; RIBEIRO, 2003).

O GRASP é um procedimento iterativo que combina uma fase construtiva e uma fase de melhoria. Na fase construtiva uma solução é construída passo a passo, adicionado elementos a uma solução parcial. A fase é iterativa, gulosa, aleatória e adaptativa. Mais adiante são descritos o procedimento construtivo utilizado, a estratégia de aleatorização que se encontra embutida no processo construtivo, os movimentos desenvolvidos para a fase de melhora, e uma fase de diversificação a qual está incluída na estrutura iterativa.

Figura 70 - Localização de uma caixa sobre um espaço vazio e os três novos espaços gerados



Fonte: Elaboração do próprio autor

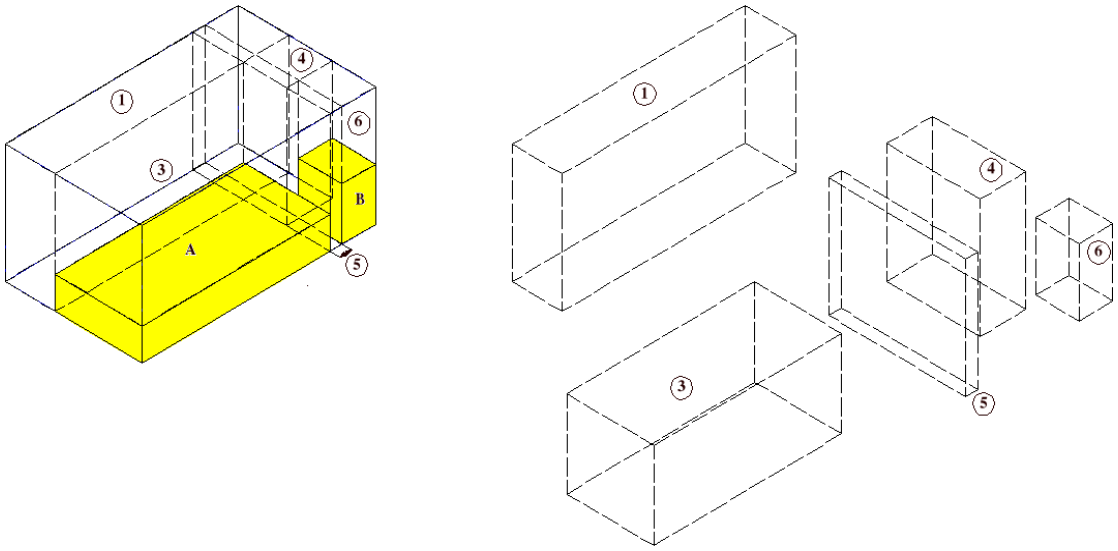
Algoritmo construtivo

O algoritmo construtivo deste trabalho está baseado na fase construtiva proposta por Parreño et al. (2010) para o problema clássico de carregamento do contêiner. O algoritmo proposto e o algoritmo de Parreño et al. (2010) se diferenciam especialmente na restrição de estabilidade vertical que neste trabalho deve ser garantida através de padrões de empacotamento com suporte completo (*full support*) e que devido o carregamento está divido em clientes deve realizar-se uma atualização dos espaços restantes e utilizáveis (vazios) segundo as definições de *multi-drop*, sendo esta tarefa a mais laboriosa visto que a gerência dos espaços vazios deixa de ser trivial e agora pode acontecer que alguns espaços devam ser eliminados ou recortados.

O algoritmo construtivo trabalha com base aos espaços máximos. Neste caso, cada caixa selecionada é empacotada num novo espaço e isto gera três novos espaços máximos (ver Figuras 70 e 71). O algoritmo construtivo usa uma lista atualizada *S* dos espaços máximos e uma lista *B* das caixas do cliente atual que ainda não foram empacotadas. Os passos do algoritmo construtivo são definidos a seguir.

Passo 0: Inicialização de *S*. É criada uma lista *S* dos espaços máximos vazios onde serão localizadas as caixas selecionadas. Seja a lista $B=\{B_1, B_2,\ldots, B_n\}$, o conjunto de caixas do cliente que restam para ser empacotadas.

Figura 71 - Procedimentos de administração dos espaços máximos: criação, atualização e eliminação

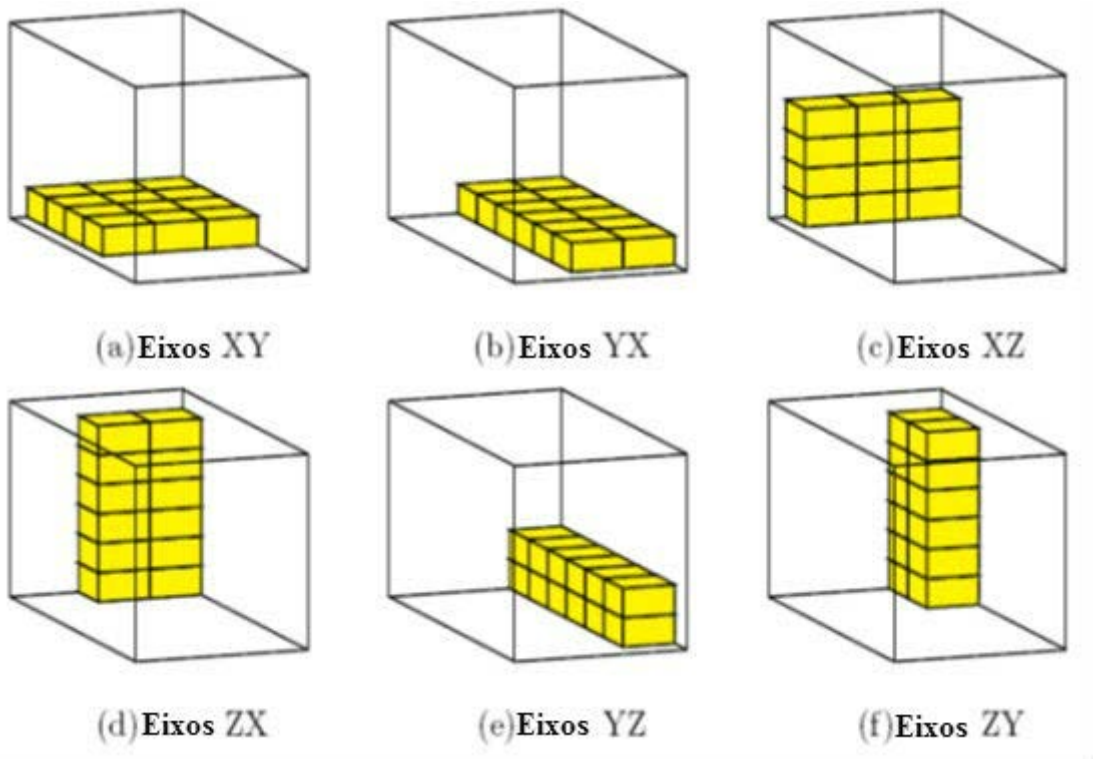


Fonte: Elaboração do próprio autor

Passo 1: Escolher um espaço máximo de *S*. A lista *S* contém os espaços máximos vazios, os quais representam os maiores paralelepípedos vazios e disponíveis para localizar as caixas. Dado que existem vários candidatos se deve determinar um mecanismo de seleção com base em algum critério de qualidade ou estratégia de empacotamento, neste trabalho se propõem dois critérios de seleção: o primeiro critério consiste em escolher o espaço máximo com mínima distância à face de trás do contêiner, já o segundo critério consiste em escolher o espaço máximo com mínima distância ao teto do contêiner. Em caso de empates entre vários candidatos se usa o critério restante para desempatar. O primeiro critério representa a ideia de ir enchendo o contêiner de trás para frente, enquanto que o segundo tenta empilhar as caixas.

Além disto, do espaço escolhido é selecionada a esquina inferior mais próxima a uma das esquinas inferiores traseiras do contêiner, como ponto de ancoragem (ou referência) na hora de localizar as caixas no espaço vazio. Este processo é diferente quando se resolve a versão do Liu et al., (2011), onde se exige a alcançabilidade por parte do montador de carga, neste caso se deve verificar como e onde pode ser localizada a caixa (ou camada de caixas) para satisfazer a restrição de atingibilidade.

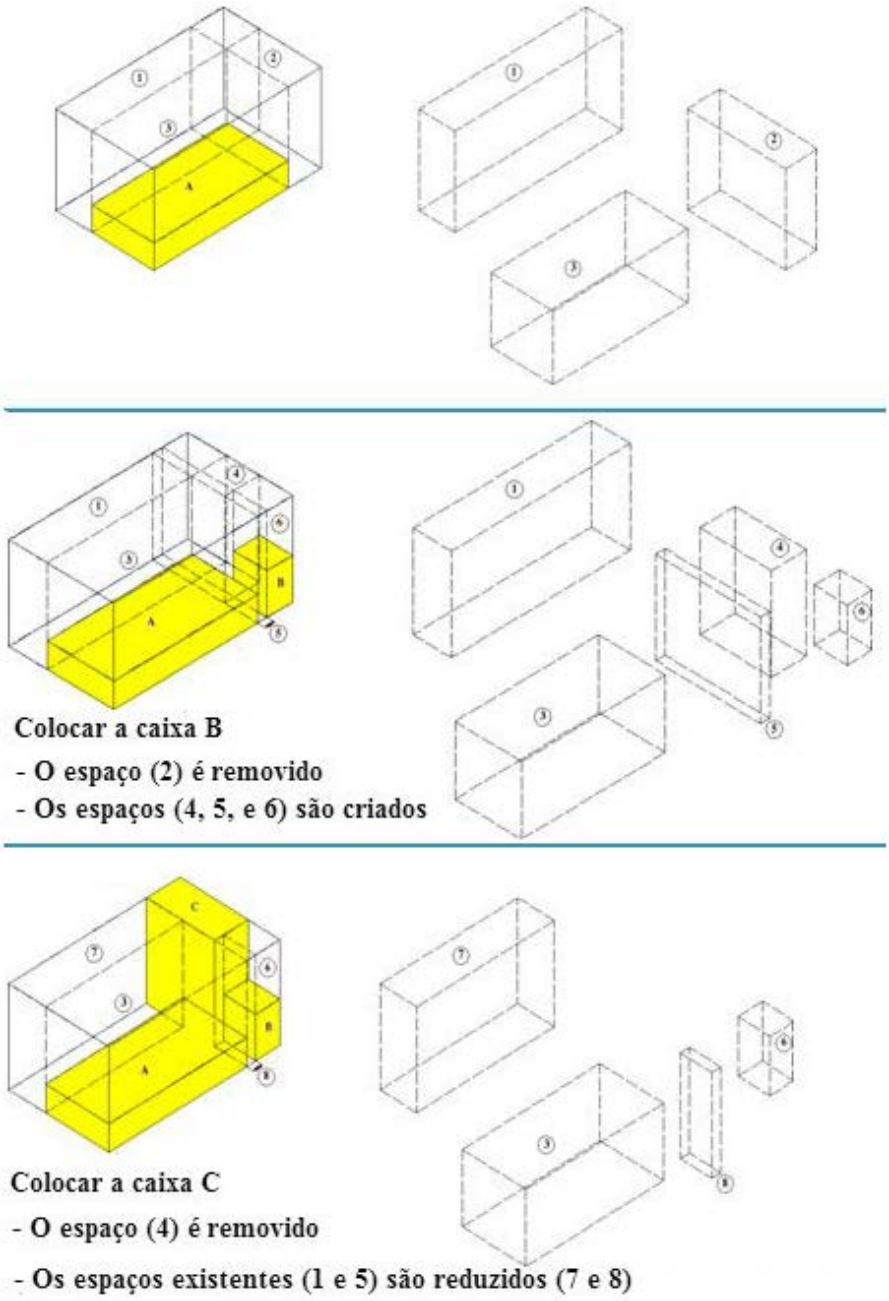
Figura 72 - Elaboração de camadas de caixas. Exemplo das seis possíveis camadas elaboradas ao combinar os distintos eixos e tendo doze exemplares do mesmo tipo de peça



Fonte: Elaboração do próprio autor

Passo 2: Escolher as caixas a empacotar. Uma vez que o espaço máximo *S'* tem sido selecionado, deve-se pegar da lista ordenada *B*, a primeira caixa *i* que encaixe dentro de *S'*. Se existem vários exemplares da caixa *i*, deve-se gerar cada uma das possíveis camadas. Isto consiste em empacotar as caixas em arranjos de colunas ou filas, combinando os diferentes eixos (primeiro em uma direção ou eixo e logo no complementar, ver Figura 72).

Figura 73 - Atualização da lista dos espaços máximos



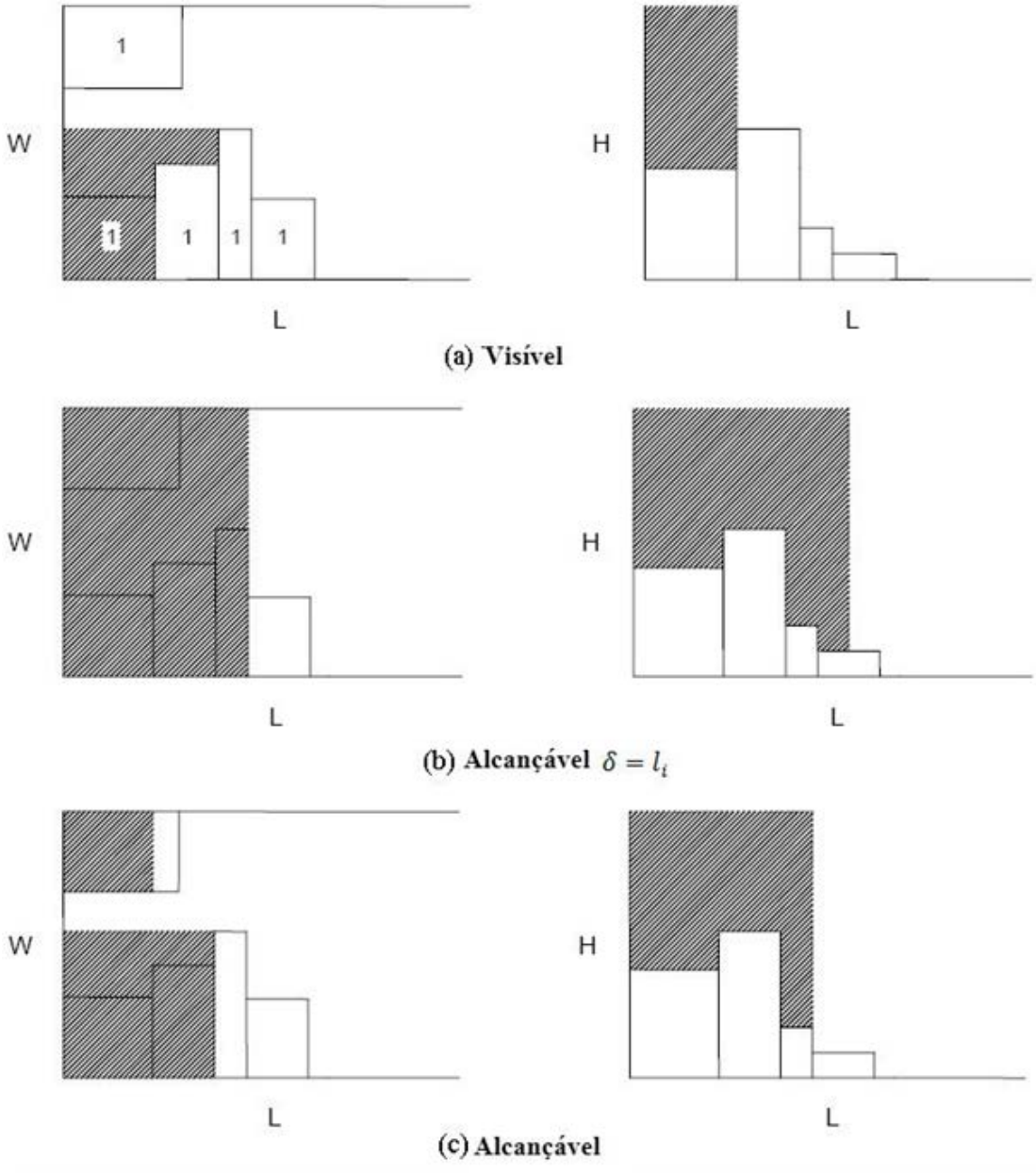
Fonte: Elaboração do próprio autor

Como em Parreño et al. (2010), dois critérios são considerados para selecionar uma das configurações das caixas:

- Escolher a camada de caixas que produz o maior incremento na função objetivo (*max volume*). Este é um critério guloso no qual o espaço é preenchido com a camada de maior volume de ocupação das caixas.

- Escolher a camada de caixas que melhor encaixam no espaço máximo (*best-fit*). Este é um critério no qual se computam as distâncias entre cada face da camada de caixas e cada face do espaço máximo e se ordenam as distâncias de cada configuração em ordem não decrescente. Para selecionar a configuração com menor distância (que melhor se ajusta ao espaço).

Figura 74 - Atualização dos espaços por troca de cliente, segundo a definição de *multi-drop*



Fonte: Elaboração do próprio autor

Passo 3: Atualizar a lista *S*. A não ser que a caixa (ou a camada) encaixe exatamente no espaço *S*, o processo de empacotar produzirá novos espaços máximos vazios que substituirá *S'* na lista *S*. Por outro lado, como os espaços máximos não são disjuntos, a caixa (ou a camada) empacotada pode interceptar-se com outros espaços máximos os quais podem ser reduzidos ou eliminados (ver Figura 73).

Uma vez os novos espaços máximos tenham sido adicionados e alguns modificados, deve ser verificada a lista *S* onde serão eliminadas as possíveis inclusões. A lista *B* é também atualizada e os espaços máximos que não possam alocar nenhuma das caixas que ainda restam por empacotar deverão ser eliminados de *S*. Se $S = \emptyset$ ou $B = \emptyset$ esta fase está terminada. Caso contrário, se ainda existem caixas por empacotar do cliente atual se deve voltar ao Passo 1.

Passo 4: Atualizar a lista *S* para um novo cliente. Quando o cliente atual tem sido empacado, os espaços máximos devem ser atualizados dependendo do tipo de critério de *multi-drop* (ver Figura 74):

- Visível. Devem ser removidos da lista todos os espaços máximos que são completamente invisíveis da porta do contêiner. Também devem ser atualizados (modificados) os espaços máximos que têm uma parte visível e outra invisível.

- Alcançáveis. Além do critério de visibilidade devem ser removidos ou atualizados todos os espaços que são inalcançáveis.

Aleatorização

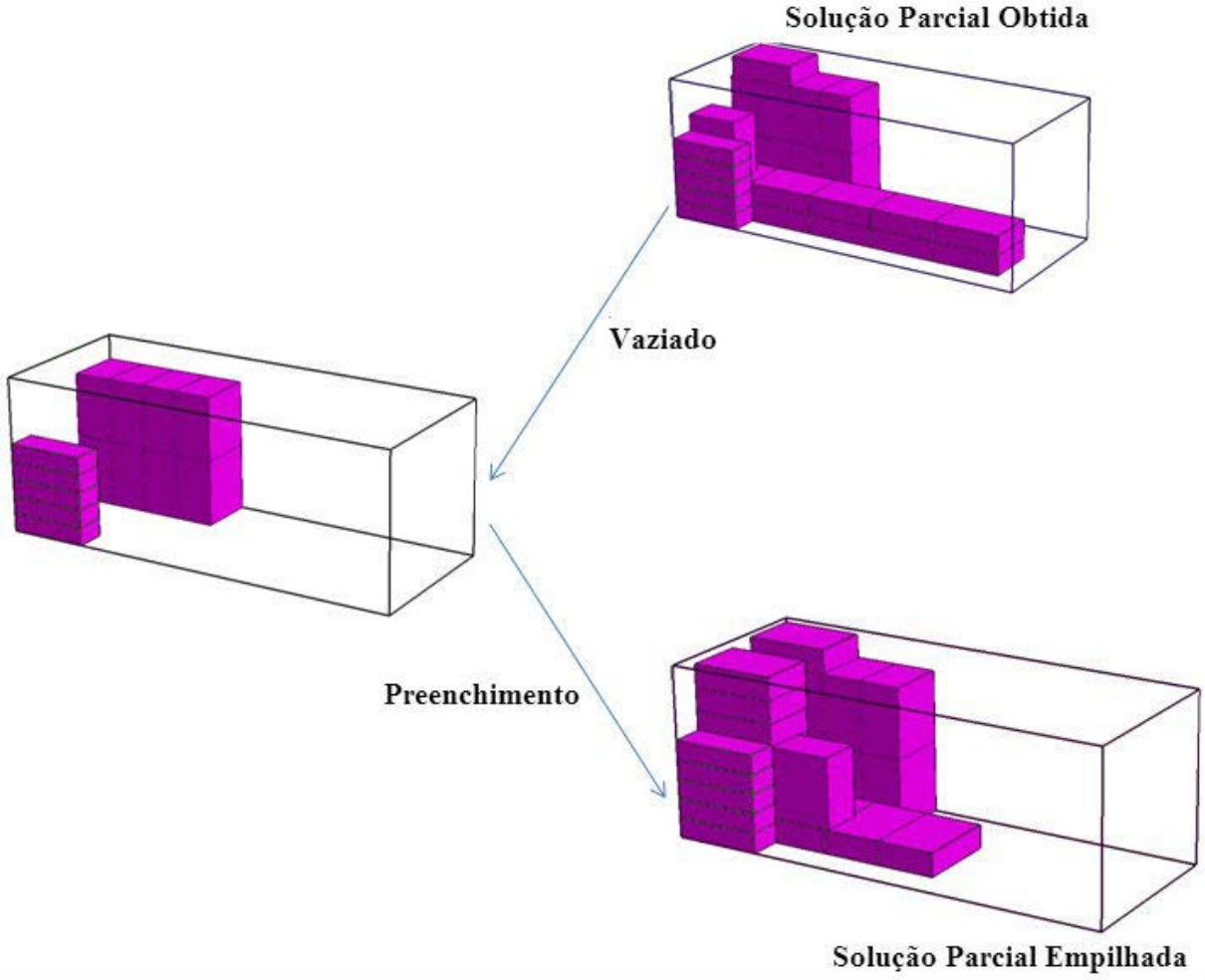
Para cada tipo de caixa e cada orientação permitida, é construída uma camada de acordo com o critério selecionado (*max volume* ou *best-fit*). Cada uma destas é chamada como configuração ou candidato. Com todo o conjunto de possíveis camadas é construída uma lista restrita de candidatos e é selecionado um destes aleatoriamente. Neste trabalho é usada uma Lista Restrita de Candidatos (RCL, do inglês *Restricted Candidate List*) segundo o valor; isso quer dizer que os candidatos são ordenados segundo seu valor de qualidade. Se o valor da função objetivo do candidato é maior que um limiar δ, o candidato é adicionado na lista. Através do processo de construção de camadas se tem um C_{min} e um C_{max}, o menor e o maior valor de todos os candidatos e o valor de cada candidato *C*. O candidato é aceito dentro do

RCL se satisfaz $C \geq C_{min} + \delta(C_{max} - C_{min})$. O parâmetro $\delta \in [0,1]$ controla o tamanho da lista de candidatos. Si $\delta = 0$, todas as configurações irão à lista e poderia ser uma seleção completamente aleatória. Em contraste, $\delta = 1$ significaria uma seleção completamente gulosa. Por causa de que só o melhor dos candidatos seria o único elemento dentro da lista e sempre será o escolhido. Para valores de $0 < \delta < 1$, o número de configurações que entram na lista não estaria predefinido, dependendo dos valores relativos dos candidatos.

Movimentos de melhora

Neste trabalho são considerados dois movimentos de melhora, um para a solução global e outro para a solução intermediária na troca de clientes.

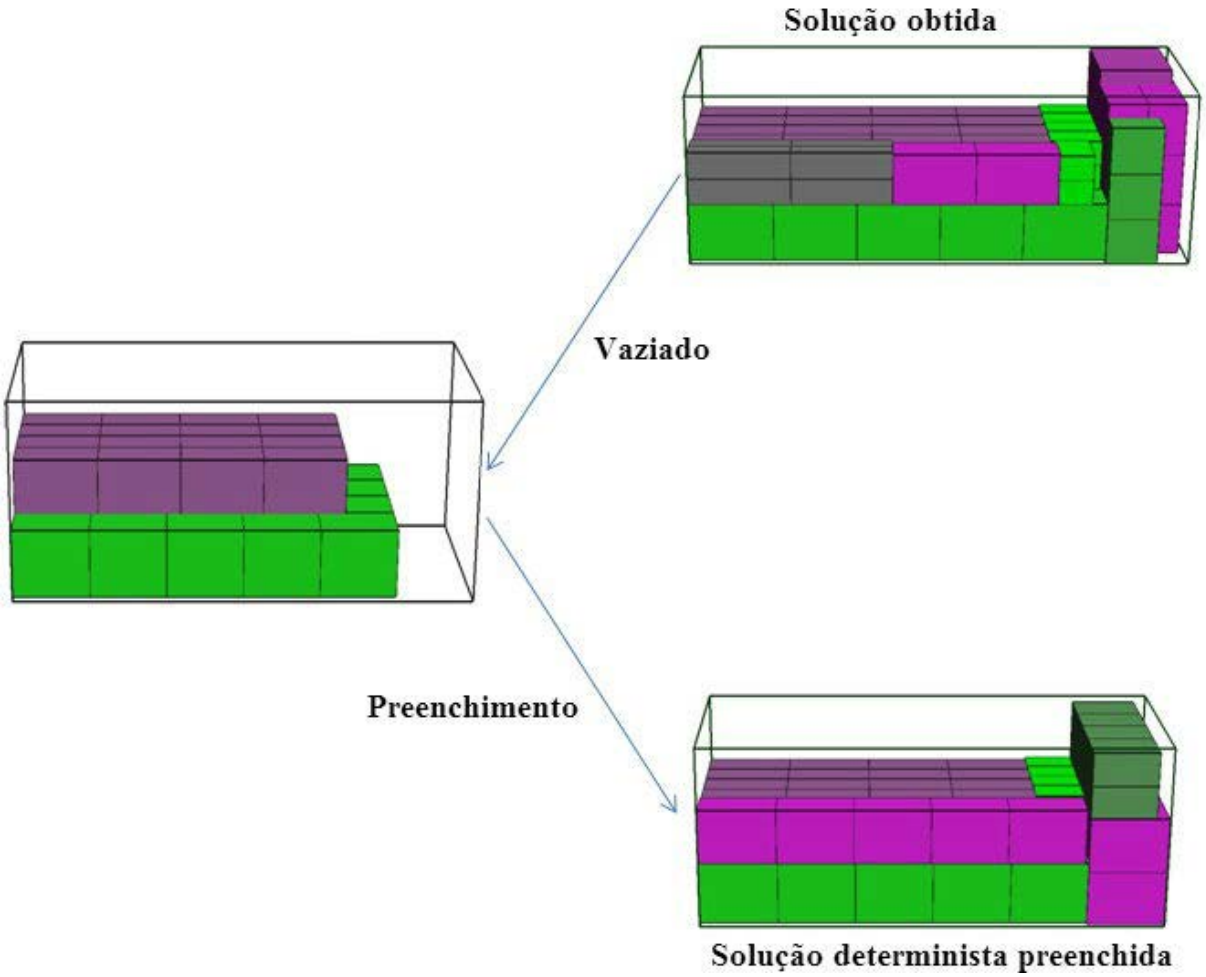
Figura 75 - Primeiro movimento de melhora. Melhora realizada ao final de cada solução parcial (troca de cliente). Esvaziado e preenchimento, priorizando soluções mais empilhadas



Fonte: Elaboração do próprio autor

O primeiro movimento consiste em aplicar a melhora anterior cada vez que se tenta realizar uma troca de cliente. Como isto poderia levar a obter as mesmas soluções, se desenvolveu uma função de avaliação da qualidade das soluções baseada na área ($W\ x\ L$) ocupada pelas caixas na solução parcial, com o objetivo de dar prioridade às soluções com um melhor índice de empilhamento (ver Figura 75).

Figura 76 - Segundo movimento de melhora. Melhora realizada ao final de obter uma solução. Esvaziado e preenchimento determinista



Fonte: Elaboração do próprio autor

O segundo movimento consiste em eliminar o último k % de caixas empacotadas da solução (por exemplo, 50%). O valor de k é escolhido aleatoriamente entre [30, 90]. As caixas removidas mais as caixas que não foram usadas são empacotadas através do algoritmo construtivo determinístico ($\delta = 1$). Neste movimento o algoritmo construtivo usa ambas as funções objetivas: configuração com maior volume e configuração que melhor encaixa (*best-fit*). Para vaziar e preencher de novo o contêiner, a solução terá que remover caixas tanto

do primeiro cliente (em ordem de descarregamento) como de clientes anteriores (ver Figura 76).

A fase de melhoramento só é chamada se a solução da fase construtiva é considerada promissora, ou seja, se esta é pode ser considerada um "bom ponto" de partida para melhorar a melhor solução obtida pelo algoritmo na iteração atual. Só são levadas em conta as soluções que estejam acima de um limiar. No começo, o limiar pega o valor da primeira solução obtida pelo algoritmo construtivo. Depois, se numa iteração o valor dessa solução ultrapassa o limiar, este deve ser atualizado e a solução entraria na fase de melhoria. Se a solução é menor que o limiar, o contador de rechaços (n_{iter}) é incrementado. Quando o número de soluções rechaçadas é maior que o número *maxFilter* (máximas rejeições), o limiar é reduzido de acordo com a Equação 9.

$threshold = threshold - \lambda(1 + threshold)$	*(9)*

Onde λ é fixada em *0.2* e *maxFilter = 5*, como sugere Parreño et al. (2010).

Fase de construção baseada na memória

Quando se resolve a variante irrestrita do problema pode modificar-se a fase de construção.

Cada elemento na fase de construção é avaliado de acordo com o volume, neste caso este valor é alternado pela combinação linear de uma função gulosa e uma função de diversificação que aumenta cada vez que um elemento aparece com mais frequência no conjunto de soluções de elite. Isto significa que algumas caixas de um cliente ou alguns clientes não serão levadas em conta na hora de empacotar. Dado que o algoritmo empacota em ordem segundo o cliente, ou seja, logo após haver empacotado o cliente *i* começará a empacotar o cliente *i+1* e assim continuará. Pode acontecer que se obtenham melhores empacotamentos se conseguimos localizar caixas que tenham ficado fora, deixando de empacotar caixas dos clientes iniciais.

Durante um número de iterações $Iter_m$ é armazenada a informação relacionada à área usada da base do contêiner de uma solução parcial. Cada vez que um cliente é atendido (empacotado) esta área é dividida pelo volume total do pedido do cliente. Para cada cliente é computada a diferença entre a área usada na iteração atual e seus predecessores, com este valor se obtém uma estimação da área da base do contêiner usada por este cliente. Esta informação é usada para selecionar os clientes e as caixas que serão melhor não levar em conta na hora de

empacotar. Vale a pena notar que é somente uma estimação de quão empilhável são as caixas deste cliente.

Figura 77 - Algoritmo GRASP

```
Inicio
    Inicializar as listas S, B, P = ∅,
    Para l = 1 até NúmeroMáximoIterações
        Enquanto B não seja ∅
            Enquanto S não seja ∅
                Espaço Si = SelecionarEspaçoSegundoCriterio(S)
                Lista Cs = GerarListaCamadas(Si, B)
                Lista RCL = ConstruirRCL(Cs, δ)
                Camada C = SelecionarAleatoriamenteCamada(RCL)
                Padrão P = LocalizarCamada-Espaco(C, Si, P)
                Lista S = AtualizarListaEspaçosMáximos(C, S)
                Lista B = AtualizarListaCaixasRestantes(C, B)
            Fim Enquanto
            Se f.o.(P)>threshold , então
                Padrão P = MelhorarPadrãoParcial(P, k)
            Caso contrário
                threshold = threshold – λ(1+ threshold)
            Fim Se
            Lista S = AtualizarListaEspaçosTrocaCliente(S, TipoMulti-drop)
        Fim Enquanto
        Se f.o.(P)>threshold , então
            Padrão P = MelhorarPadrãoFinal(P, k)
        Caso contrário
            threshold = threshold – λ(1+ threshold)
        Fim Se
    Seguinte l
Fim
```

Fonte: Elaboração do próprio autor

Portanto, para cada cliente temos um indicador e quanto maior é este número significa que seu pedido vai requerer de maior área do piso do contêiner para ser empacotado. Com base neste, se remove uma percentagem de caixas para as próximas iterações. A percentagem depende da qualidade da melhor solução atual obtida (MSAO) pelo algoritmo, é removida ao máximo um valor aleatório entre (0.3*(Volume total – Volume MSAO), 0.7*(Volume total – Volume MSAO)). Este valor é ajustado pela metade da diferença entre o volume total demandado da instância do problema e o volume da melhor solução atual obtida. Isto significa que nas próximas iterações se resolverá o problema com menor volume demandado e a ordem de empacotamento pode ter sido alterada (no caso de que todas as caixas de um cliente tenham sido removidas).

É selecionado um cliente com o maior valor e se removem aleatoriamente caixas deste cliente até ser alcançada a percentagem predefinida, se esta percentagem não é alcançada removendo as caixas de um cliente, se passa ao seguinte cliente com pior indicador e assim continua.
Na figura 77 ilustra-se um pseudocódigo do algoritmo GRASP proposto, neste mostra-se através do uso de listas os passos de seleção do espaço, criação das camadas, seleção aleatória de uma das camadas, localização da camada no espaço, construção do padrão, melhoria do padrão parcial e melhoria do padrão final.

4.5 RESULTADOS COMPUTACIONAIS

O algoritmo desenvolvido foi codificado em C++ e executado usando uma máquina com as seguintes descrições: um processador Intel Core i7-2600 ® com 3.40 GHz e uma memória RAM de 12GB. Desenvolveu-se um estudo computacional extenso utilizando casos de estudo e problemas reais apresentados na literatura (ver Tabela 7). Como ponto de referência, são usados os casos de teste gerados pelo Bischoff e Ratcliff (1995). Estes 700 problemas são divididos em sete classes (BR1,…, BR7) cada uma com 100 instâncias, o número de tipos de caixas em cada classe vai desde 3 até 20 tipos diferentes (BR7, a sétima classe). Além disto, neste trabalho se estende até o uso das classes BR8-BR15 que apresenta instancias com demanda de peças fortemente heterogêneas, desenvolvidas também pelo Bischoff e Ratcliff (1995). O número de tipos diferentes de caixas está num rango de 30 a 100 (BR8 a BR15, respectivamente).

Nestes casos as caixas foram geradas independentemente das dimensões do contêiner, pelo que não é garantido que todas as caixas demandadas de uma instância podam ser colocadas dentro do contêiner. O peso de cada caixa é proporcional ao seu volume, ou seja, a densidade de todo o material da caixa se assume igual a *1*. Os limites de empilhamento para cada face da caixa são definidos aleatoriamente usando uma distribuição normal. Todas as instâncias estão disponíveis na página Web do ESICUP: http://paginas.fe.up.pt/~esicup/tiki-index.php?page_ref_id=6 (acessado em fevereiro 3, de 2014).

Adicionalmente, para realizar estudos sobre a restrição de *multi-drop*, Christensen e Rousøe (2009) fazendo uso das 700 instancias (BR1-BR7) definem um número de clientes (2, 5, 10 o 50) e atribuí cada caixa a um cliente usando uma função aleatória uniforme. A atribuição é

realizada independentemente para cada número de clientes, isto acaba gerando assim 3500 instâncias para o estudo do problema *multi-drop*.

Tabela 7 - Detalhes do benchmarking

Trabalho	**Restrições**						**Instancias**		
	Orientação das peças	Limites de empilhamento	Limites de peso	Estabilidade do carregamento	*Multi-drop*	Envio completo[d]	Quantidade	Descrição	Fonte
Liu, et al. (2011)	X			X	X	U2	100	thpack7	Bischoff e Ratcliff (1995)[a]
Junqueira, et al. (2012a)	X			X	X	U2	8	*Randomly generated instances*	Junqueira, et al. (2012a)
							8	*Real-world instances*	Christensen e Rousøe (2009)[b]
Christensen e Rousøe (2009)	X			X	X	R	3500	thpack1-7	Bischoff e Ratcliff (1995)[c]
Ceschia e Schaerf (2013)	X	X	X	X	X	R	117	*Real-world instances*	Ceschia e Schaerf (2013)

(a) Conjunto de instâncias adaptadas para incluir restrições *multi-drop*
(b) Instâncias adaptadas para não incluir restrições de limite de empilhamento das caixas
(c) Conjunto de 500 instancias adaptadas para incluir *multi-drop* e assim gerar 3500 instâncias
(d) Envio completo: R, restrito; U1, irrestrito tipo 1; U2, irrestrito tipo 2.

Neste estudo, o algoritmo proposto é comparado com o procedimento exato desenvolvido por Junqueira et al. (2012a) usando 16 instâncias que eles gerarem. Também são usadas as instâncias apresentadas por Ceschia e Schaerf (2013), estas últimas foram filtradas para usar unicamente as instâncias com um único contêiner, estes casos de teste se encontram disponíveis na página Web: http://satt.diegm.uniud.it/3Dpacking/ (acessada em junho 14, de 2013).

Por último, são usados os cem (100) problemas propostos pelo Liu et al. (2011), a atribuição utilizada neste estudo é diferente à usada pelos autores, devido a que seus dados não se encontram disponíveis. No entanto, seguindo o processo de elaboração proposto pelos autores, é criado um conjunto de 100 problemas e são reportados unicamente os valores médios, apesar de que não é possível eliminar a inclinação nos resultados. As instancias

geradas utilizam a classe BR7, porém neste caso cada exemplar de um diferente tipo de peça é atribuído a um cliente aleatório.

Tabela 8 - Comparação de resultados obtidos pelo algoritmo GRASP proposto contra a definição e algoritmo de Ceschia e Schaerf (2013)

Observações	**Nome da Instância**	$\frac{VolumenPacked}{L \cdot W \cdot H}$	
		Ceschia e Schaerf	**GRASP**
Todas as caixas são empacotadas. Por causa da baixa taxa de volume para empacotar.	CS2843	44,54	44,54
	CS3048	48,37	48,37
	CS3152	69,85	69,85
	CS3182	77,15	77,15
	CS3291	69,82	69,82
	CS3388	36,38	36,38
	CS3556	42,83	42,83
	CS3695	41,50	41,50
	CS3941	70,26	70,26
	CS2000	79,08	86,66
	CS2805	67,37	69,85
	CS2822	73,03	75,62
	CS2899	79,00	85,67
	CS3056	50,17	60,70
	CS3074	71,00	75,31
	CS3122	67,31	69,18
	CS3142	71,66	75,93
	CS3151	65,02	70,06
	CS3203	83,72	86,21
	CS3207	67,59	72,75
	CS3314	80,42	80,42
	CS3432	75,60	80,16
	CS3915	49,58	62,02
Média		70,04	75,04
Tempo computacional médio (seg)			18,66

Neste trabalho se fixa um número máximo de iterações igual a 50.000. Em todas as Tabelas apresentadas a seguir (Tabelas 8-14) a média corresponde à percentagem de volume utilizado.

Comparação com outros algoritmos heurísticos (versão restrita)

Primeiro se apresenta na Tabela 8 a comparação com Ceschia e Schaerf (2013) por causa de que é o trabalho com mais restrições práticas levadas em conta, para isto são separadas 23 instâncias das 117 propostas, dado que somente se está utilizando um único contêiner.

Tabela 9 - Comparação de resultados obtidos pelo algoritmo GRASP proposto contra a definição e algoritmo de Christensen e Rousøe (2009)

Número de clientes	**Conjunto de instâncias (100 casos por grupo)**								
	GRASP								**Christensen e Rousøe**
	BR1	**BR2**	**BR3**	**BR4**	**BR5**	**BR6**	**BR7**	**Média**	**Média**
1	92,47	92,75	92.76	92.22	91.68	91.01	89.62	91.79	89.07
2	91,77	90,76	89.47	88.56	87.78	86.69	85.51	88.65	85.96
5	88,56	86,02	83.15	81.36	80.14	78.30	76.12	81.95	78.52
10	85,28	81,43	77.82	76.14	74.47	72.45	70.22	76.83	72.64
50	79,32	73,96	68.98	66.87	64.85	63.25	60.61	68.26	64.45
Média	87,48	84,98	82.44	81.03	79.78	78.34	76.42	81.50	78.13
Tempo computacional médio (seg)								47,81	60

Na Tabela 9 se mostram os resultados do algoritmo proposto comparado com o algoritmo de Christensen e Rousøe (2009). Seu algoritmo foi programado em C++ e todos os testes foram executados numa máquina Linux com um processador AMD® de 2.4GHz 64 bits e uma memória RAM de 2GB. Eles têm ajustado um tempo limite de 60 segundos.

A Tabela 9 mostra que os resultados obtidos pelo algoritmo proposto são competitivos com os melhores resultados reportados para esta classe de problemas. Mostra-se que não somente a qualidade da solução depende do número de clientes como também do número de tipos de caixas. Quando existem 3 tipos de caixas (BR1) a qualidade da solução cai um 13% de 1 a 50 clientes. Quando o número de tipos de caixas existentes é 20 (BR7) a diferença chega a ser dos 30%. Os problemas mais difíceis são os que têm muitos clientes e muitos tipos de caixas.

Tabela 10 - Comparação de resultados obtidos pelo algoritmo GRASP proposto contra a definição, e algoritmo de Liu et al. (2011) e o algoritmo SBIP apresentado por Jin et al. (2004)

Instâncias BR7 - Modificadas	$\frac{VolumenPacked}{L \cdot W \cdot H}$		
	SBIP - LIU	**Alg 3.1 - Liu**	**GRASP**
Média	49,90	48,78	**49,52**
Tempo médio (seg)	63,76	51,98	84,11

Para esta versão se comparou com os resultados reportados por Liu et al. (2011) e o algoritmo SBIP apresentado por Jin et al. (2004). Estes executarem seus teste sobre uma máquina com processador Intel Core 2 Duo E8400® de 3.00 GHz e com uma memória RAM de 2 GB. A Tabela 10 mostra que o algoritmo proposto é superior ao de Liu et al. (2011), mas inferior à implementação do algoritmo de Jin et al. É importante ressaltar as baixas percentagens de volume de utilização, isto é por causa de que se está resolvendo a versão 3D-SKP (*three-dimensional single knapsack problem*), já que a demanda de peças é fortemente heterogênea (um exemplar por tipo de caixa e cliente).

Envios completos irrestritos

Nas Tabelas 11 e 12 são mostrados os resultados da versão do algoritmo na qual não são empacotadas em ordem todas as caixas de cada cliente. Primeiro se realiza a comparação com o algoritmo exato apresentado por Junqueira et al. (2012a), os testes deste foram executadas numa máquina com um processador Intel Core i7® de 2.8 GHz e com uma memória RAM de 8.0 GB, foi fixado um tempo limite de 3600 segundos. Estes gerarem um conjunto de instancias, com elas provam ambos os valores do parâmetro δ_i ($\delta_i = 0$ *e* $\delta_i = l_i$).

Esta comparação (a versão irrestrita do Junqueira et al. (2012a)) é realizada por duas razões. Primeiro porque o trabalho deles representa outra definição do *multi-drop* e segundo porque é um modelo exato e permite verificar as soluções obtidas pelo algoritmo heurístico proposto neste trabalho. Nestas instâncias foi fixado em 3 o número de destinos.

Tabela 11 - Comparação de resultados obtidos pelo algoritmo GRASP proposto contra a formulação e definição de Junqueira et al., com $\delta_{ik} = 0$

$\delta_{ik} = 0$						
Nome	**Nome do arquivo**	**Número de caixas**	**Número de tipos**	**L**	$\frac{VolumenPacked}{\sum_{i=1}^{n} x_i \cdot y_i \cdot z_i}$	
					Junqueira	**GRASP**
A1	6_n10m01b2d2_05md	20	1	15	1	1
		20	1	12	0,95	0,95
A5	1_n10m05b2d2_02md	41	5	15	1	1
		41	5	12	0,779	0,955
A10	1_n10m10b2d2_04md	99	10	15	1	1
A20	1_n10m20b2d2_09md	89	20	15	1	1
		89	20	12	0,928	0,984
B1	6_n10m01b2d1_07md	50	1	15	1	1
B5	1_n10m05b2d1_10md	813	5	15	1	1
B10	1_n10m10b2d1_06md	1000	10	15	1	1
B20	1_n10m20b2d1_05md	674	20	15	1	1
Tempo computacional médio (seg)					42,75	5,85

Tabela 12 - Comparação de resultados obtidos pelo algoritmo GRASP proposto contra a formulação e definição de Junqueira et al., com $\delta_{ik} = l_i$

$\delta_{ik} = l_i$						
Nome	**Nome do arquivo**	**Número de caixas**	**Número de Tipos**	**L**	$\frac{VolumenPacked}{\sum_{i=1}^{n} x_i \cdot y_i \cdot z_i}$	
					Junqueira	**GRASP**
A1	6_n10m01b2d2_05md	20	1	15	1	1
A5	1_n10m05b2d2_02md	41	5	15	1	1
		41	5	12	0,9	1
A10	1_n10m10b2d2_04md	99	10	15	1	1
A20	1_n10m20b2d2_09md	89	20	15	1	1
B1	6_n10m01b2d1_07md	50	1	15	1	1
B5	1_n10m05b2d1_10md	813	5	15	1	1
B10	1_n10m10b2d1_06md	1000	10	15	1	1
B20	1_n10m20b2d1_05md	674	20	15	1	1
Tempo computacional médio (seg)					863,45	10,31

A Tabela 13 mostra os resultados apresentados por Liu et al. (2011). Estes propõem dois algoritmos (algoritmos 3.1 e 3.2) em seu trabalho, no segundo algoritmo os autores relaxam a restrição de “empacotar caixas na ordem estrita” e assim estes podem localizar caixas que não estão na ordem previamente estabelecida.

Tabela 13 - Comparação de resultados obtidos pelo algoritmo GRASP proposto contra a definição, e o algoritmo 3.2 de Liu et al. (2011)

Instancias BR7 - Modificadas	$\frac{VolumenPacked}{L \cdot W \cdot H}$	
	Alg 3.2 - Liu	**GRASP**
Média	53,18	53,27
Tempo médio (seg)	51,98	**84,11**

Tabela 14 - Comparação de resultados obtidos pelo algoritmo GRASP proposto contra o algoritmo de Alonso et al. (2014)

Classe	**GRASP**	**Alonso et al.**	**Classe**	**GRASP**	**Alonso et al.**
BR1	82,52	81,4	**BR8**	**83,74**	85,5
BR2	86,66	85,7	**BR9**	**82,41**	84,8
BR3	88,42	87,3	**BR10**	**81,38**	84,0
BR4	88,10	86,9	**BR11**	**80,07**	82,8
BR5	87,40	86,6	**BR12**	**78,88**	81,3
BR6	86,91	86,3	**BR13**	**77,65**	80,2
BR7	**85,53**	85,7	**BR14**	**76,58**	78,9
			BR15	**75,48**	78,0
Média (BR1-BR7)	86,51	85,7	**Média (BR8-BR15)**	**79,52**	83,7
Tempo médio (seg)	**13,5**	9,8	**Tempo médio (seg)**	**102,8**	64,7
Média (BR1-BR15)	**82,8**	83,5			

Restrições de limite de resistência das caixas ao empilhamento com um único cliente

Embora as características principais do algoritmo proposto tentem explorar ao máximo as restrições de *multi-drop*, este também pode ser comparado com outros algoritmos propostos na literatura, como o algoritmo de Alonso et al. (2014) para resolver a variante do problema

de carregamento do contêiner com restrições de limite de resistência das caixas ao empilhamento. Para isto são usados os mesmos casos de teste apresentados em (BISCHOFF; RAFCLIFF, 1995).

Alonso et al. (2014) apresentam um algoritmo GRASP que utiliza diferentes funções objetivos, enquanto que sua representação de espaços vazios está baseada na proposta pelo Ngoi (1994). Estes executaram seus testes numa máquina com um processador Intel Core Duo T6500 ® de 2.1GHz e uma memória RAM de 4GB. A Tabela 14 mostra que os resultados obtidos pelo algoritmo proposto neste trabalho são de boa qualidade, já que se está comparando contra a melhor referência na literatura e adicionalmente este é um algoritmo adaptado para essa variante do problema. Deve-se notar que quanto mais heterogêneas são as instâncias o comportamento do algoritmo proposto diminui de qualidade.

4.6 CONCLUSÕES

Se desenvolveu uma metodologia de solução para o problema de carregamento de contêineres, que usa a representação dos espaços máximos para administrar os espaços residuais, o processo de otimização é gerenciado por um algoritmo construtivo aleatorizado, que permite gerar soluções factíveis e consegue ser flexível e robusto para solucionar as diferentes definições de empacotamento *multi-drop* na literatura.

Resolver variantes do problema de carregamento de contêineres envolvendo restrições e condições reais representa um alto grau de dificuldade não somente pela complexidade inata dos problemas de empacotamento como também por causa do pouco consenso da comunidade acadêmica e as diferentes interpretações que cada autor baseado na sua perícia dá às características do problema.

O algoritmo proposto é flexível e pode ser adaptado a outras restrições de empacotamento como distribuição do peso do carregamento dentro do contêiner e empacotamento de caixas com peso não proporcional a seu volume (a densidade do material é um parâmetro do sistema). Adicionalmente, o algoritmo é combinável com problemas como o problema de roteamento de veículos com restrições de empacotamento tridimensional.

5 CONCLUSÕES E TRABALHOS FUTUROS

Nesta pesquisa indicamos diferentes algoritmos de otimização metaheurística para resolver os problemas de corte ótimo na mochila bidimensional (2D-SLOPP, do inglês Two-Dimensional *Single Large Object Placement Problem*,) com restrições de padrão de corte, valores associadas às peças, limites de exemplares por tipo peça e orientação das peças. O segundo problema estudado foi, o problema da embalagem (2D-SBSBPP, do inglês *Two-Dimensional Single Bin Size Bin Packing Problem*) com restrições de padrão de corte e orientação das peças. E por último, o problema do carregamento de um único contêiner (3D-SLOPP, do inglês *Three-Dimensional Single Large Object Placement Problem*) com restrições de orientação das caixas, limites de resistência de pesos das caixas ao empilhamento, limite do peso do carregamento suportado pelo contêiner, estabilidade da carga e carregamento fracionado em múltiplos destinos.

Realizou-se a revisão bibliográfica dos modelos matemáticos dos problemas de corte e empacotamento ótimo bidimensional e tridimensional guilhotinado e não guilhotinado. Os problemas estudados nesta tese têm sido analisados por mais de seis décadas porém ainda não se chegou a um consenso para determinar um modelo matemático bem definido e que incluía as diferentes características que forneçam soluções a situações práticas do problema na vida real. Em especial para os problemas de corte se encontram deficiências ao adicionar as restrições padrões de corte tipo guilhotina em etapas, enquanto os problemas de empacotamento, as formulações matemáticas apresentam fraquezas ao adicionar as restrições de carga com múltiplos destinos.

Para os problemas de corte foram utilizados os conceitos de três técnicas meta-heurísticas (otimização por enxame de partículas, busca em vizinhança variável e algoritmos genéticos) e com base nesses se programou um algoritmo híbrido para resolver os problemas da mochila bidimensional e o problema da embalagem. Para o problema de carregamento de um único contêiner, foram utilizados os conceitos da técnica metaheurística GRASP para administrar um algoritmo construtivo aleatório com processos de melhoramento.

Foram revisadas diferentes codificações e representações propostas na literatura para resolver os problemas de corte bidimensional. Assim, foi selecionada a codificação em árvore de cortes e a representação dos espaços máximos para a manipulação dos espaços residuais.

Enquanto que, para os problemas de empacotamento foi adaptada a representação dos espaços máximos em três dimensões para cumprir com as restrições do problema.

A análise de resultados realizado nesta investigação demonstra o desempenho dos algoritmos propostos para resolver os casos teste da literatura especializada (apresentados para os problemas deste trabalho). Tanto em tempos computacionais, como em qualidade de resultados.

Trabalhos futuros propostos nesta investigação

Ampliar a representação utilizada dos espaços máximos no estudo problemas de empacotamento ótimo tridimensional, para levá-la ao problema de empacotamento ótimo de caixas em contêineres usando *pallets*.

Usar outras técnicas meta-heurísticas de otimização utilizando a codificação e a representação proposta com o fim de melhorar a qualidade dos resultados.

Utilizar uma metodologia exata para realizar o ajuste ótimo de parâmetros das técnicas meta-heurísticas de otimização propostas neste trabalho.

Aplicar esta metodologia de solução em problemas da vida real, o qual se consegue através do uso de informação real apresentada na Indústria.

REFERÊNCIAS

ALONSO, M. T.; ALVAREZ-VALDES, R.; TAMARIT, J. M. PARREÑO, F. A reactive GRASP algorithm for the container loading problem with load-bearing constraints. **European Journal of Industrial Engineering**, Bucks, 2014. (No prelo)

ÁLVAREZ, D.; TORO, E. M.; GALLEGO, R. A. Problema de la mochila irrestricta bidimensional guillotinada. **Revista Ingeniería & Universidad**, Cali, v. 14, n. 2, p. 327-344, 2010.

ÁLVAREZ, D.; TORO, E. M.; GALLEGO, R. A. Estudio comparativo de algoritmos basados en cúmulo de partículas para resolver el problema de empaquetamiento en placas. **Revista Ingeniería y Competitividad**, Cali, v. 13, n. 1, p. 113-130, 2011.

ÁLVAREZ, D.; ESCOBAR, L. M.; ROMERO, R. A. Equipo asíncrono de agentes basados en recocido simulado aplicado al problema del agente viajero simétrico. **Scientia et Technica**, Pereira, v. 49, p. 122-127, 2011.

ALVAREZ-VALDÉS, R.; PARAJÓN, A.; TAMARIT, J. M. A tabu search algorithm for large-scale guillotine unconstrained two-dimensional cutting problems. **Computers & Operations Research**, New York, v. 29, p. 925–947, 2002.

ALVAREZ-VALDÉS, R.; PARREÑO F.; TAMARIT J. M. A tabu search algorithm for a twodimensional non-guillotine cutting problem. **European Journal of Operational Research**, Amsterdam, v. 183, p. 1167–1182, 2007.

ANDREWS, P. S. An investigation into mutation operators for particle swarm optimization. In: IEEE CONGRESS EVOLUTION COMPUTATION, Vancouver, 2006. **Proceedings of the…** Vancouver, p. 1044–1051, 2006. Disponível em: < http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1688424>. Acesso em: 17 abr. 2013.

ARENALES, M.; MORABITO, R.; YANASSE, H. Cutting and packing problems. **Pesquisa Operacional**, Rio de Janeiro, v. 19, n. 2, p. 107-299, 1999.

BALDACCI, R.; BOSCHETTI, M. A. A cutting-plane approach for the two dimensional orthogonal non-guillotine cutting problem. **European Journal of Operational Research,** Amsterdam, v. 1833, p. 1136-1149, 2007.

BEASLEY, J. E. Algorithms for unconstrained two-dimensional guillotine cutting. **Journal Operations Research Society of Amarica**, Baltimore, v. 36, p. 297–306, 1985.

BEASLEY, J. E. An exact two-dimensional non-guillotine cutting tree-search procedure. **Operations Research**, Baltimore, v. 33 p. 49–64, 1986.

BEASLEY, J. E. A population heuristic for constrained two-dimensional nonguillotine cutting. **European Journal of Operational Research**, Amsterdam, v. 156, p. 601- 627, 2004.

BEKRAR, A.; KACEM, I.; CHU, C.; SADFI, C. An improved heuristic and an exact algorithm for the 2D strip and bin packing problema. **International Journal Product Development**, Bucks, v. 10, n. 1/3, p. 217-240, 2010.

BEN, S.; CHU, C.; ESPINOUSE, M. L. Characterization and modelling of guillotine constraints. **European Journal of Operational Research**, Amsterdam, v. 191, p. 112–126, 2008.

BERKEY, J. O.; WANG, P. Y. Two-dimensional finite bin packing algorithms. **Journal of the Operational Research Society**, Baltimore, v. 38, p. 423–429, 1987.

BINKLEY K. B.; HAGIWARA, M. Applying self-adaptive evolutionary algorithms to two-dimensional packing problems using a four corners' heurístic. **European Journal of Operational Research**, Amsterdam, v. 183, p. 1230–1248, 2007.

BIRGIN, E. G.; LOBATO, R. D. Orthogonal packing of identical rectangles within isotropic convex regions. **Computers & Industrial Engineering,** New York, v. 59, n. 4, p. 595-602, 2010.

BIRGIN, E. G.; LOBATO, R. D.; MORABITO, R. An effective recursive partitioning approach for the packing of identical rectangles in a rectangle, **Journal of the Operational Research Society,** Amsterdam, v. 61, n. 2, p. 306-320, 2010.

BIRGIN, E. G.; LOBATO, R. D.; MORABITO, R. Generating unconstrained two-dimensional non-guillotine cutting patterns by a recursive partitioning algorithm. **Journal of the Operational Research Society,** Baltimore, v. 63, p. 183-200, 2012.

BIRGIN, E. G.; MARTÍNEZ, J. M.; MASCARENHAS, W. F.; RONCONI, D. P. Method of sentinels for packing items within arbitrary convex regions. **Journal of the Operational Research Society,** Amsterdam, v. 57, n. 6, p. 735-746, 2006a.

BIRGIN, E. G.; MARTÍNEZ, J. M.; NISHIHARA, F. H. RONCONI, D. P. Orthogonal packing of rectangular items within arbitrary convex regions by nonlinear optimization. **Computers & Operations Research,** New York, v. 33, n. 12, p. 3535-3548, 2006b.

BIRGIN, E. G.; MORABITO, R.; NISHIHARA, F. H. A note on an L-approach for solving the manufacturer's pallet loading problema. **Journal of the Operational Research Society,** Amsterdam, v. 56, n. 12, p. 1448-1451, 2005.

BIRO, M.; BOROS, E. Network flows and non-guillotine cutting patterns. **European Journal of Operational Research**, Amsterdam, v. 16, p. 215–221, 1984.

BISCHOFF, E. E. Three-dimensional packing of items with limited load bearing strength. **European Journal of Operational Research**, Amstedam, v. 168, p. 952-966, 2006.

BISCHOFF, E. E.; MARRIOTT, M. D. A comparative evaluation of heuristics for container loading. **European Journal of Operational Research**, Amsterdam, v. 44, p. 267–276, 1990.

BISCHOFF, E. E.; RATCLIFF, M. S. W. Issues in the development of approaches to container loading. **Omega**, Elmsford, v. 234, p. 377–390, 1995.

BISCHOFF, E. E.; WÄSCHER, G. Cutting and packing. **European Journal of Operational Research**, Amsterdam, v. 84, n. 3, p. 503-505, 1995.

BORTFELDT, A. A hybrid algorithm for the capacitated vehicle routing problem with three-dimensional loading constraints. **European Journal of Operational Research**, Amsterdam, v. 399, p. 2248-2257, 2012.

BORTFELDT, A.; GEHRING, H. A hybrid genetic algorithm for the container loading problem. **European Journal of Operational Research**, Amsterdam, v. 131, p. 143–161, 2001.

BORTFELDT, A.; WÄSCHER, G. Constraints in container loading - a state-of-the-art review. **European Journal of Operational Research**, Amsterdam, v. 2291, p. 1-20, 2013.

BORTFELDT, A.; WINTER, T. A genetic algorithm for the two-dimensional knapsack problem with rectangular pieces. **International Transactions in Operational Research**, Oxford, v. 16, p. 685–713, 2009.

BOSCHETTI M. A.; MINGOZZI A. ;HADJICONSTANTINOU, E. New upper bounds for the two dimensional orthogonal non-guillotine cutting stock problem. **IMA Journal of Management Mathematics**, Oxford, v. 13, p. 95–119, 2002.

BROWN, A. R. **Optimum packing and depletion: the computer in space - and resource usage problems**. London: Macdonald and Co, 1971. 107 p.

CAPRARA, A.; MONACI, M. On the two-dimensional knapsack problem. **Operations Research Letters**, Amsterdam, v. 32, p. 5–14, 2004.

CARLISLE, A.; DOZIER, G. Adapting particle swarm optimization to dynamic environments. In: INTERNATIONAL CONFERENCE ARTIFICIAL INTELLIGENT, **Proceedings of the…** Las Vegas: IEEE, 2000. p. 429–434.

CESCHIA, S.; SCHAERF, A. Local search for a multi-drop multi-container loading problem. **Journal of Heuristics**, New York, v. 19, n. 2, p. 275-294, 2013.

CESCHIA, S.; SCHAERF, A.; STÛTZLE, T. Local search techniques for a routing-packing problema. **Computers and Industrial Engineering**, Oxford, v. 66, n. 4, p. 1138-1149, 2013. Disponível em: < http://ac.els-cdn.com/S0360835213002404/1-s2.0-S0360835213002404-main.pdf?_tid=31112974-df92-11e3-bc2c-00000aab0f6c&acdnat=1400530653_c1264991e0fef171bf12ce47974c0522>. Acesso em: 19 mar. 2014.

CHEN, M.; HUANG, W. A two-level search algorithm for 2D rectangular packing problem, **Computers & Industrial Engineering,** New York, v. 53, p. 123–136, 2007.

CHEN, C. S.; LEE, S. M.; SHEN Q. S. An analytical model for the container loading problem. **European Journal of Operational Research**, Amsterdam, v. 80, p. 68–76, 1995.

CHENG, C. H.; FEIRING, B. R.; CHENG, T. C. E. The cutting stock problem - a survey. **International Journal of Production Economics**, Amsterdam, v. 36, n. 3, p. 291-305, 1994.

CHRISTENSEN, S. G.; ROUSØE, D. M. Container loading with multi-drop constraints. **International Transactions in Operational Research**, Amsterdam, v. 16, p. 727-743, 2009.

CHRISTOFIDES, N. Optimal cutting of two-dimensional rectangular plates. In: CAD 74, INTERNATIONAL CONFERENCE ON COMPUTERS IN ENGINEERING AND BUILDING DESIGN, 1974, London. **Proceedings of the…** London: Imperial College, 1974. Disponível em: < https://getinfo.de/app/CAD-74-International-Conference-on-Computers-in/id/TIBKAT%3A596220847>. Acesso em: 14 abr. 2014.

CHRISTOFIDES, N.; WHITLOCK, C. An algorithm for two-dimensional cutting problems. **Operations Research**, Baltimore, v. 25, p. 30–44, 1977

CHARALAMBOUS, C.; FLESZAR, K. A constructive bin-oriented heuristic for the two-dimensional bin packing problem with guillotine cuts. **Computers & Operations Research**, New York, v. 38, n. 10, p. 1443–1451, 2011.

CINTRA, G. F.; MIYAZAWA, F. K.; WAKABAYASHI, Y.; XAVIER E. C. Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation. **European Journal of Operational Research**, Amsterdam, v. 191, p. 61–85, 2008

COFFMAN, E.; GAREY, M.; JOHNSON, D. Approximation algorithms for bin packing: a survey. In: HOCHBAUM, D. S. **Approximation algorithms for NP-hard problems.** Boston: PWS Publishing, 1996. p. 46-93.

CUI, Y. An exact algorithm for generating homogenous two-segment cutting patterns. **Engineering Optimization**, London, v. 39, p, 365–380, 2007a.

CUI, Y. An exact algorithm for generating homogenous T-shape cutting patterns. **Computers & Operations Research**, New York, v. 34, p. 1107-1120, 2007b.

CUI, Y. Heuristic and exact algorithms for generating homogenous constrained three-staged cutting patterns. **Computers & Operations Research**, New York, v. 35, p. 212–225, 2008.

DANIELS, K.; MILENKOVIC, V. J.; LI, Z. **Multiple containement methods**. Harvard: Center for Research in Computing Technology, Division of Applied Sciences, 1994. (Technical Report 12-94)

DE CASTRO SILVA, J. L.; SOMA, N. Y.; MACULAN, N. A greedy search for the tree-dimensional bin packing problem: the packing stability case. Internat. **Transactions in Operational Research**, Oxford, v. 10, p. 141–153, 2003.

DELL'AMICO, M.; MARTELLO, S.; VIGO, D. A lower bound for the non-oriented two-dimensional bin packing problem. **Discrete Applied Mathematics,** Amsterdam, v. 118, p. 13–24. 2002.

DOWSLAND, K. Some experiments with simulated annealing techniques for packing problems. **European Journal of Operational Research**, Amsterdam, v. 68, p. 389–399, 1993.

DOWSLAND, W. B. Two and three dimensional packing problems and solution methods**.** **New Zealand Operational Research**, Wellington, v. 13, n. 1, p. 1-18, 1985.

DOWSLAND, K. A.; DOWSLAND, W. B. Packing problems. **European Journal of Operational Research**, Amsterdam, v. 56, n. 1, p. 2-14, 1992.

DUECK, G.; SCHEUER, T. Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. **Journal of Computational Physics**, Orlando, v. 90, p. 161–175, 1990.

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

DYCKHOFF, H.; FINKE, U. **Cutting and packing in production and distribution:** typology and bibliography. Heidelberg: Springer-Verlag, 1992. 248 p.

DYCKHOFF, H.; KRUSE, H. J.; ABEL, D.; GAL, T. Trim loss and related problems. **Omega**, Elmsford, v. 13, n. 1, p. 59-72, 1985.

DYCKHOFF, H.; SCHEITHAUER, G.; TERNO, J. Cutting and packing. In: DELL'AMICO, M.; MAFFIOLI, F.; MARTELLO, S. **Annotated bibliographies in combinatorial optimization.** New York: John Wiley & Sonsp, 1997. p. 393-414.

DYSON, R. G.; GREGORY, A. S. The cutting stock problem in the flat glass industry. **Operational Research Quartely,** London, v. 25, p. 41-53. 1976.

EGEBLAD, J.; PISINGER, D. Heuristic approaches for the two- and three-dimensional knapsack packing problem. **Computers and Operations Research**, New York, v. 36, p. 1026–1049, 2009.

FAROE, O.; PISINGER, D.; ZACHARIASEN, M. Guided local search for the three-dimensional bin-packing problem. **INFORMS Journal on Computing**, Linthicum, v. 15, n. 3, p. 267283, 2003.

FARLEY, A. A. Mathematical programming model for cutting stock problems in the clothing industry. **Journal of the Operation Operation Research Society,** Amsterdam, v. 39, p. 41-53, 1988.

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

FASANO, G. Cargo Analytical integration in space engineering: a three-dimensional packing model. In: CIRIANI, T. A. et al. (Ed.). **Operational research in industry**. Purdue: University Press, 1999. p. 232-246.

FAYARD, D.; HIFI, M.; ZISSIMOPOULOS, V. An efficient approach for large-scale two-dimensional guillotine cutting stock problems. **Journal of the Operational Research Society,** Amsterdam, v. 49, p. 1270–1277, 1998

FAYARD, D.; ZISSIMOPOULOS, V. An approximation Algorithm For Solving Unconstrained Two-Space Knapsack problems. **European Journal of Operations Research**, Amsterdam, v. 84, p. 618-632, 1995.

FEKETE, S.; SCHEPERS, J. **On more-dimensional packing III:** exact algorithms., Mathematisches Institut, UniversitYat zu KYoln, 1997. (Technical Report ZPR97-290).

FEKETE, S.; SCHEPERS, J.; VANDER VEEN J. An exact algorithm for higher-dimensional orthogonal packing. **Operations Research**, Baltimore, v. 55, p. 569–587, 2007.

FEO, T.; RESENDE, M. G. C. A probabilistic heuristic for a computationally difficult set covering problem. **Operations Research Letters**, Amsterdam, v. 8, p. 67-71, 1989.

FLESZAR, K. Three insertion heuristics and a justification improvement heuristic for two-dimensional bin packing with guillotine cuts. **Computers & Operations Research**, New York, v. 40, n. 1, p. 463-474, 2013.

GAREY, M. R.; JOHNSON, D. S. **Computers and intractability:** a guide to the theory of NP-completeness. San Francisco: W. H. Freeman, 1979.

GENDREAU, M.; IORI, M.; LAPORTE G.; MARTELLO, S. A tabu search algorithm for a routing and container loading problema. **Transportation Science**, v. 403, p. 342-350, 2006.

GILMORE, P. C.; GOMORY, R. E. A linear programming approach to the cutting stock problema. **Operations Research**, Elmsford, v. 9, p. 849-859, 1961.

GILMORE, P. C.; GOMORY, R. E. A linear programming approach to the cutting stock problem – part II. **Operations Research**, Baltimore, v. 11, p. 863–888, 1963.

GILMORE, P. C.; GOMORY, R. E. Multistage cutting problems of two and more dimensions. **Operations Research**, Baltimore, v. 13, p. 94-120, 1965.

GILMORE, P. C.; GOMORY, R. E. The theory and computation of knapsack functions. **Operations Research**, Baltimore, v. 14, p. 1045-1074, 1966.

GOLDEN, B. Approaches to the cutting stock problem. **AIIE Transactions**, New York, v. 8, n. 2, p. 265-274, 1976.

GONÇALVES, J. F. A hybrid genetic algorithm-heuristic for a two-dimensional orthogonal packing problema. **European Journal of Operational Research,** Amsterdam, v. 183, p. 1212-1229, 2007.

HADJICONSTANTINOU, E.; CHRISTOFIDES, N. An exact algorithm for general, orthogonal, two-dimensional knapsack problems. **European Journal of Operational Research,** Amsterdam, v. 83, n. 1, p. 39-56, 1995.

HAESSLER, R. W.; SWEENEY, P. E. Cutting stock problems and solution procedures. **European Journal of Operational Research**, Amsterdam, v. 54, n. 2, p. 141-150, 1991.

HAESSLER, R. W.; TALBOT F. B. Load planning for shipments of low density products. **European Journal of Operational Research**, Amsterdam, v. 44, n. 2, p. 289-299, 1990.

HERZ, J. C. A recursive computing procedure for two-dimensional stock cutting. I.B.M. **Journal Research Development**, Bingley, v. 16, 462-469, 1972

HIFI, M. An improvement of Viswanathan and Bagchis exact algorithm for constrained two-dimensional cutting stock. **Computers & Operations Research**, New York, v. 24,p. 727–736, 1997a

HIFI, M. **Problem instances for the 2D Cutting/Packing Problems**. 1997b. Disponível em: <ftp://cermsem.univ-paris1.fr/pub/CERMSEM/hifi/2Dcutting/>. Acesso em: 15 abr. 2014.

HIFI, M. Exact algorithms for large-scale unconstrained two and three staged cutting problems. **Computational Optimization and Applications**. Hinghan, v. 18, p. 63–88, 2001.

HIFI, M. Cutting and packing problems. **Studia Informatica Universalis**, [S.l], v. 2, n. 1, p. 1-161, 2002.

HIFI, M.; M'HALLAH, R. An exact algorithm for constrained two-dimensional two-staged cutting problems. **Operations Research,** Baltimore, v. 531, p. 140-150, 2005.

HIFI, M.; ROUCAIROL, C. Approximate and exact algorithms for constrained unweighted two-dimensional two-staged cutting stock problems. **Journal of Combinatorial Optimization,** Boston, v. 5, p. 465–494, 2001

HIFI, M.; ZISSIMOPOULOS, V. A recursive exact algorithm for weighted two-dimensional cutting. **European Journal Operations Research**, Amsterdam, v. 91, p. 553–564, 1996.

HINXMAN, A. I. The trim-loss and assortment problems: a survey. **European Journal of Operational Research**, Amsterdam, v. 5, n. 1, p. 8-18, 1980.

HOPPER, E.; TURTON, B. C. H. A review of the application of meta-heuristic algorithms to 2D strip packing problems. **Artificial Intelligence Review**, Dordrecht, v. 16, p. 257–300, 2001a.

HOPPER, E.; TURTON, B. C. H. An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem, **European Journal of Operational Research**, Amsterdam, v; 128, p. 34–57, 2001b.

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

JIN, Z.; OHNO, K.; DU, J. An efficient approach for the three-dimensional container packing problem with practical constraints. **Asia-Pacific Journal of Operational Research**, Singapore, v. 21, n. 3, p. 279-295, 2004.

JUNQUEIRA, L.; MORABITO, R.; YAMASHITA, D. S. MIP-based approaches for the container loading problem with multi-drop constraints. **Annals Operations Research,** Amsterdam, v. 199, p. 51–75, 2012a.

JUNQUEIRA, L.; MORABITO, R.; YAMASHITA, D. S. Three-dimensional container loading models with cargo stability and load bearing constraints. **Computers & Operations Research**, New York, v. 39, n. 1, p. 74-85, 2012b.

JUNQUEIRA, L.; OLIVEIRA, J. F.; CARRAVILLA, M. A.; MORABITO, R. An optimization model for the vehicle routing problem with practical three-dimensional loading; **International Transactions in Operational Research,** Oxford, v. 20, n. 5, p. 645-666, 2013.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: IEEE INTERNATIONAL CONFERENCE ON NEURAL NETWORKS, 4, LEARNING AND CYBERNETICS. **Proceedings...** 2002, Beijing. 2002. p. 1942-1948, 1995. Disponível em: < http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=488968>. Acesso em: 10 abr. 2014.

KENNEDY, J.; EBERHART, R. C. A discrete binary version of the particle swarm algorithm. In: CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS, 1997, Piscataway. **Proceedings of the...** Piscataway: IEEE Service Center, 1997. p. 4104-4109, 1997.

KENNEDY, J.; EBERHART, R.C. **Swarm intelligence**. San Francisco: Morgan Kaufmann, 2001. 512 p.

KIRKPATRICK, S.; GELATT, C.; VECCHI, M. Optimization by simulated annealing. **Science**, Washington, v. 220, p. 671–680, 1983.

KRÖGER, B. Guillotineable bin packing: a genetic approach. **European Journal of Operational Research**, Amsterdam, v. 84, p. 645-661, 1995.

LAI, K. K.; CHAN, J. W. M. A evolutionary algorithm for the rectangular cutting stock problem. **International Journal on Industrial Engineering**, New York, v. 4, p. 130–139, 1997.

LEUNG, T. W.; YUNG, C. H.; TROUTT, M. D. Application of mixed simulated annealing-genetic algorithm heuristic for the two-dimensional orthogonal packing problem. **European Journal of Operational Research,** Amsterdam, v. 145, p. 530–542, 2003.

LIANG, J. J.; SUGANTHAN, P. N. Dynamic multi-swarm particle swarm optimizer with local search. In: THE CONGRESS IEEE EVOLUTIONARY COMPUTATION, 2005, Edinburgh. **Proceedings of the...** Edinburgh: IEEE, 2005.p. 522–528, 2005. Disponível em: < http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1554727>. Acesso em: 15 abr. 2014.

LINS, L.; LINS, S.; MORABITO, R. An L-approach for packing (*l*, *w*)-rectangles into rectangular and L-shaped pieces. **Journal of the Operational Research Society,** Amsterdam, v. 54, n. 7, p. 777-789, 2003.

LIU, W. Y.; LIN, C. C.; YU, C. S. On the three-dimensional container packing problem under home delivery service. **Asia-Pacific Journal of Operational Research**, Singapore, v. 28, n.5, p. 601-621, 2011. Disponível em: < http://www.worldscientific.com/doi/pdf/10.1142/S0217595911003466>. Acesso em: 10 jan. 2014.

LODI, A.; MARTELLO S.; MONACI, M. Two-dimensional packing problems: a survey. **European Journal of Operational Research**, Amsterdam, v. 141, p. 241–252, 2002.

LODI, A.; MARTELLO S.; VIGO, D. **Problem instances for the two-dimensional bin packing problema.** 1997. Disponível em: <http://www.or.deis.unibo.it/research_pages/ORinstances/2BP.html>. Acesso em: 10 mar. 2014.

LODI, A.; MARTELLO S.; VIGO, D. Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. **INFORMS Journal Computing,** Linthicum, v. 11, p. 345–357, 1999.

LODI, A.; MARTELLO S.; VIGO, D. Models and bounds for the two-dimensional level packing problems. **Journal of Combinatorial Optimization**, Boston, v. 8, p. 363–379, 2004.

MADSEN O. G. B. Glass cutting in small firm. **Mathematical Programming,** Amsterdam, v. 17, p. 85-90, 1979.

MARTELLO, S. Knapsack, packing and cutting, Part I: one-dimensional knapsack problems. **INFOR**, Otawa, v. 32, n. 3, 1994a.

MARTELLO, S. Knapsack, packing and cutting, Part II: multidimensional knapsack and cutting stock problems. **INFOR**, Otawa, v. 32, n. 4, 1994b.

MARTELLO, S.; TOTH, P. **Knapsack problems:** algorithms and computer implementations. West Sussex: John Wiley & Sons, 1990. 296 p.

MARTELLO, S.; MONACI, M.; VIGO, D. An exact approach to the strip-packing problem, **INFORMS Journal on Computing**, Linthicum, v. 15, n. 3, p. 310–319, 2003.

MARTINS, G. H. A. **Packing in two and three dimensions**. 2003. Thesis (Doctor)- Naval Postgraduate School, Monterey, California, 2003.

MASCARENHAS, W. F.; BIRGIN, E. G. Using sentinels to detect intersections of convex and nonconvex polygons. **Computational & Applied Mathematics,** Antwerpen, v. 29, n. 2, 247-267, 2010.

MLADENOVIĆ, N.; HANSEN, P. Variable neighborhood search. **Computers and Operations Research,** Amsterdam, v. 24, p. 1097–1100, 1997.

MORABITO, R.; ARENALES, M. Um exame dos problemas de corte e empacotamento. **Pesquisa Operacional**, Rio de Janeiro, v. 12, n. 1, p. 1-20, 1992.

MORABITO, R.; ARENALES, M. Staged and constrained two-dimensional guillotine cutting problems: an AND/OR-graph approach. **European Journal of Operational Research,** Amsterdam, v. 94, n. 3, p. 548–560, 1996.

MORABITO, R.; ARENALES, M.; ARCARO, V. F. An and/or-graph approach for two-dimensional cutting problems. **European Journal of Operational Research,** Amsterdam, v. 58, n. 2, p. 263-271, 1992.

MORABITO, R.; GARCIA, V. The cutting stock problem in a hardboard industry: a case study. **Computers & Operations Research,** Amsterdam, v. 25, n. 6, p. 469-485, 1997.

MORABITO, R.; MORALES, S. A simple and effective recursive procedure for the manufacturer's pallet loading problema. **Journal of the Operational Research Society**, Amsterdam, v. 49, p. 819-828, 1998.

MORABITO, R.; MORALES, S. A simple and effective recursive procedure for the manufacturer's pallet loading problema. **Journal of the Operational Research Society**, v. 50, p. 876 - 876, 1999.

MORABITO, M.; PUREZA, V. Generation of constrained two-dimensional guillotine cutting patterns via dynamic programming and and/or-graph search. **Produção,** São Paulo, v. 17, p. 33–51, 2007.

NGOI, B. K. A.; TAY, M. L.; CHUA, E. S. Applying spatial representation techniques to the container packing problema. **International Journal of Production Research**, London, v. 32, p. 111-123, 1994.

OLIVEIRA, J. F.; WÄSCHER, G. Cutting and packing. **European Journal of Operational Research**, Amsterdam, v. 183, n. 3, p. 1106-1108, 2007.

ONODERA, H.; TANIGUCHI, Y.; TAMARU, K. Branch-and-bound placement for building block layout. In: ACM/IEEE DESIGN AUTOMATION CONFERENCE, 28, **Proceedings**

of the... 1991. p. 433–439. Disponível em: < http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=979755>. Acesso em: 10 mar. 2013.

PADBERG, M. Packing small boxes into a big box. **Mathematical Methods of Operations Research**, Heidelberg, v. 52, n. 1, p. 1-21, 2000.

PARREÑO, F.; ALVAREZ-VALDES, R.; TAMARIT, J. M.; OLIVEIRA, J. F. A maximal-space algorithm for the container loading problem. **INFORMS Journal on Computing,** Linthicum, v. 20, n. 3, p. 412-422, 2008.

PARREÑO, F.; ALVAREZ-VALDES, R.; TAMARIT, J. M.; OLIVEIRA, J. F. Neighborhood structures for the container loading problem: a VNS implementation. **Journal of Heuristics**, Boston, v. 161, p. 1-22, 2010.

PUCHINGER, J.; RAIDL, G. R. Models and algorithms for three-stage two-dimensional bin packing. **European Journal of Operational Research**, Amsterdam, v. 183, p. 1304–1327, 2006.

RAM, B.; The pallet loading problem: a survey. **International Journal of Production Economics**, Amsterdam, v. 28, n. 2, p. 217-225, 1992.

RESENDE, M. G. C.; RIBEIRO, C. C. Greedy randomized adaptive search procedures. In GLOVER, F.; KOCHENBERGER, G. (Ed.). **Handbook of metaheuristics.** Boston: Kluwer, 2003. p. 219-249.

SCHEITHAUER, G.; TERNO, J. Modeling of packing problems. **Optimization**, Berlin, v. 28, p. 63-84, 1993.

SCHEITHAUER, G. LP-based bounds for the container and multi-container loading problem. **International Transactions in Operations Research**, Amsterdam, v. 6, p. 199–213, 1998.

SILVA, E.; ALVELOS, F.; VALERIO DE CARVALHO, J. M. An integer programming model for two- and three-stage two-dimensional cutting stock problems. **European Journal of Operational Research,** Amsterdam, v. 205, n. 3, p. 699-708. 2010.

SHI, Y.; EBERHART, R. C. Parameter selection in particle swarm optimization. **Lecture Notes in Computer Science**, Heidelberg, v. 1447, p 591-600, 1998.

SONG, X.; CHU, C.B.; LEWIS, R.; NIE, Y.Y.; THOMPSON J. A worst case analysis of a dynamic programming-based heuristic algorithm for 2D unconstrained guillotine cutting, **European Journal of Operational Research**, Amsterdam, v. 202, p. 368–378, 2010.

SWEENEY, P.; PATERNOSTER, R. Cutting and packing problems: a categorized, application-orientated research bibliography. **Journal of the Operational Research Society**, Amsterdam, v. 43, p. 691-706, 1992.

TARANTILIS, C.; ZACHARIDIS, E. E.; KIRANOUDIS, C. T. A hybrid metaheuristic algorithm for the integrated vehicle routing and three-dimensional container-loading problem. **IEEE Transactions on Intelligent Transportation Systems**, New York, v. 102, p. 255-271, 2009.

TORO, E.; GARCÉS, A.; RUIZ, H. Solución al problema de empaquetamiento bidimensional usando un algoritmo híbrido constructivo de búsqueda en vecindad variable recocido

simulado. **Revista Facultad de Ingeniería Universidad de Antioquia**, Medellin, v. 46, p. 119-131, 2008.

TSAI, R. D.; MALSTROM, E. M.; KUO, W. Three dimensional palletization of mixed box sizes. **IIE Transactions**, New York, v. 25, n. 4, p. 64-75, 1993.

TSCHÖKE, S.; HOLTHÖFER, N. **A new parallel approach to the constrained two-dimensional cutting stock problem**. Padeborn: University Padeborn, 1996. (Technical Report, University of Padeborn, D.C.S. 33095).

VENKATESWARLU, P.; MARTYN, C. W. The trim loss problem in a wooden drum industry. In: CONVENTION OF OPERATION RESEARCH SOCIETY OF INDIA. **Proceedings...** 1992. Disponível em: < http://www.thebookshelf.auckland.ac.nz/docs/NZOperationalResearch/conferenceproceedings/1992-proceedings/ORSNZ-proceedings-1992-20.pdf>. Acesso em: 12 jan. 2014.

VIGO, D.; MARTELLO, S. Exac solution of the two-dimensional finite bin packing problema. **Management Science**, Providence, v. 44, p. 388-399, 1998.

VISWANATHAN, K.V.; BAGCHI, A. Best-first search methods for constrained two-dimensional cutting stock problems. **Operations Research,** Baltimore, v. 41, p. 768–776, 1993.

WANG, P. Y. Two algorithms for constrained two-dimensional cutting stock problems. **Operations Research**, Baltimore, v. 32, p. 573–586 1983.

WANG, P. Y.; WÄSCHER, G. Cutting and packing. **European Journal of Operational Research**, Amsterdam, v. 141, n. 2, p. 239-240, 2002.

WÄSCHER, G.; HAUSSNER, H.; SCHUMANN, H. An improved typology of cutting and packing problems. **European Journal of Operational Research**, Amsterdam, v. 183, n. 3, p. 1109-1130, 2007.

WEI, L.; ZHANG, D.; CHEN, Q. A least wasted first heuristic algorithm for the rectangular packing problem. **Computers & Operations Research,** New York, v. 365, p. 1608–1614, 2009.

WESTERNLUND, T.; ISAKSOON J.; HARJUNKOSKI I. **Solving a production optimization problem in the paper industry**. Abo: Abo Akademi University 1995. (Report 95-146ª).

WONG, D. F.; LEONG, H. W.; LIU, C. L. **Simulated annealing for VLSI design**. New York: Kluwer Academic Publishers, 1988.

WU, Y. An effective quasi-human based heuristic for solving the rectangle packing problem. **European Journal of Operational Research,** Amsterdam, v. 141, p. 341–358, 2002.

WY, J.; KIM, B. Two-staged guillotine cut, two-dimensional bin packing optimization with flexible bin sizes for steel mother plate design. **International Journal of Production Research**, London, v. 4822, p. 6799–6820. 2010.

YOUNG-GUN, G.; KANG, M. K. A new upper bound for unconstrained two-dimensional cutting and packing. **Journal Operations Research Society**, Amsterdam, v. 53, p. 587–591, 2002.

YOUNG-GUN, G.; KANG, M. K.; SEONG, J. A best-first branch and bound algorithm for unconstrained two-dimensional cutting problems. **Operations Research Letters**, Amsterdam, v. 31, p. 301–307, 2003.

ZHAN, Z.; ZHANG, J.; Li, Y.; CHUNG, H. S. H. Adaptive particle swarm optimization. **IEEE Transactions on System, Man and Cybernetics – B**, New York, v. 39, p. 1362–1381, 2009.

ZHU, W.; LIM, A. A new iterative-doubling Greedy–Lookahead algorithm for the single container loading problem, **European Journal of Operational Research,** Amsterdam, v. 222, p. 408–417, 2012.

ZHU, W.; OON, W. C.; LIM, A.; WENG, Y. The six elements to block-building approaches for the single container loading problem. **Applied Intelligence**, Dordrecht, v. 37, n. 3, p. 431–445, 2012.

ANEXO

TABELAS DE RESULTADOS DO PROBLEMA DA MOCHILA

Tabela 15 - Resultados para o problema *unconstrained non-staged weighted fixed*

ID	Unconstrained non-staged Weighted Fixed Problem			
	MSP	**Tempo (seg)**	$A_{PSO+VNS+TF}$	**Tempo (seg)**
W1	162.867	2,01[a]	162.867	25
W2	35.159	0,21[a]	35.159	25
W3	234.108	8,90[a]	234.108	37
UW1	6036	0,01[b]	6036	29
UW2	8468	0,03[a]	8468	33
UW3	6302	0,02[b]	6302	28
UW4	8326	0,01[b]	8326	39
UW5	7780	0,04[b]	7780	54
UW6	6615	0,04[b]	6615	43
UW7	10.464	0,09[b]	10.464	57
UW8	7692	0,14[b]	7692	60
UW9	7038	0,23[b]	7038	68
UW10	7507	0,30[b]	7507	61
UW11	15.747	0,03[a]	15.747	33

(MSP) Melhores Soluções Publicadas
(a) DPH *algorithm,* Song et al. (2010);
(b) Young-Gun et al. (2003).

Tabela 16 - Resultados para o problema *unconstrained non-staged unweigthed fixed*

ID	Unconstrained non-staged Unweigthed Fixed Problem			
	MSP	**Tempo (seg)**	$A_{PSO+VNS+TF}$	**Tempo (seg)**
GCUT1	90,34	0[a]	90,34	9
GCUT2	96,86	0[a]	96,86	17
GCUT3	97,66	0[a]	97,66	31
GCUT4	98,72	0[a]	98,72	48
GCUT5	98,40	0[a]	98,40	11
GCUT6	95,60	0[a]	95,60	23
GCUT7	97,03	0[a]	97,03	29
GCUT8	98,65	0[a]	98,65	53
GCUT9	97,11	0[a]	97,11	12
GCUT10	98,20	0[a]	98,20	20
GCUT11	98,01	0[a]	98,01	31
GCUT12	98,00	0[a]	98,00	49
GCUT13	99,98	1,5[b]	99,98	50

(a) SDP *Algorithm*, Cintra et al. (2008);
(b) DPH *algorithm,* Song et al. (2010).

Tabela 17 - Resultados para o problema *unconstrained non-staged unweigthed rotated*

ID	Unconstrained non-staged Unweigthed Rotated Problem			
	MSP*	**Tempo (seg)**	$A_{PSO+VNS+TF}$	**Tempo (seg)**
GCUT1	93,02	0	93,02	16
GCUT2	96,98	0	96,98	30
GCUT3	98,60	0	98,60	55
GCUT4	99,62	0,01	99,62	87
GCUT5	98,40	0	98,40	19
GCUT6	96,38	0	96,38	39
GCUT7	98,35	0	98,35	53
GCUT8	99,11	0,02	99,11	97
GCUT9	97,11	0	97,11	15
GCUT10	98,20	0	98,20	40
GCUT11	98,01	0,02	98,01	56
GCUT12	98,87	0,03	98,87	91
GCUT13	100	110,20	100	100

(*) Todas as Melhores Soluções Publicadas (MSP) são do algoritmo SDP de Cintra et al. (2008).

Tabela 18 - Resultados para o problema *constrained non-staged weighted fixed*

ID	Constrained non-staged Weighted Fixed Problem			
	MSP	**Tempo (seg)**	$\mathbf{A_{PSO+VNS+TF}}$	**Tempo (seg)**
CW1	6402	5[a]	6402	74
CW2	5354	7[a]	5354	68
CW3	5689	27[b]	**5687**	104
CW4	6175	47[b]	6175	94
CW5	11.644	1269[b]	11.644	100
CW6	12.923	891[b]	**12.651**	162
CW7	9898	11[c]	9898	134
CW8	4605	341[b]	**4504**	184
CW9	10.748	121[b]	10,748	144
CW10	6515	182[b]	6515	142
CW11	6321	1375[b]	6321	124
TH1	4620	N/A[d]	4620	86
TH2	9700	N/A[d]	**9560**	54
CHW1	2892	N/A[d]	**2730**	26
CHW2	1860	N/A[d]	1860	68

(a) FHZ98 *algorithm*, Fayard et al. (1998);
(b) MP_AOG *algorithm*, Morabito e Pureza, (2007);
(c) MP *algorithm*, Morabito e Pureza, (2007);
(d) ATP2 *algorithm*, Álvarez-Valdés et al. (2002).

Tabela 19 - Resultados para o problema *constrained non-staged weighted fixed*

ID	Constrained non-staged Unweighted Fixed Problem			
	MSP	**Tempo (seg)**	**$A_{PSO+VNS+TF}$**	**Tempo (seg)**
CU1	98,64	<1[a]	98,64	101
CU2	99,43	<1[a]	99,43	105
CU3	99,84	6[b]	**99,58**	201
CU4	98,62	35[b]	**98,49**	181
CU5	98,75	768[a]	98,75	190
CU6	99,65	10[a]	99,65	252
CU7	95,85	1800[b]	95,85	276
CU8	97,65	21[a]	**97,51**	269
CU9	97,78	20[b]	97,78	243
CU10	98,94	1800[b]	**98,90**	241
CU11	99,31	1293[b]	**99,04**	470

(a) MP_AOG *algorithm*, Morabito e Pureza, (2007);
(b) MP *algorithm*, Morabito e Pureza, (2007).

Tabela 20 - Resultados para o problema *unconstrained two-staged rotated unweighted*

Unconstrained Two-staged Rotated Unweigthed Problem				
ID	**MSP***	**Tempo (seg)**	$A_{PSO+VNS+TF}$	**Tempo (seg)**
UU1	98,42	2	98,42	50
UU2	99,21	1	99,21	56
UU3	98,41	4	98,41	50
UU4	99,05	7	99,05	71
UU5	99,64	11	99,64	92
UU6	98,81	11	98,81	72
UU7	99,01	14	99,01	98
UU8	98,98	1	98,98	106
UU9	99,32	22	99,32	116
UU10	99,16	30	99,16	106
UU11	99,49	27	99,49	56
LUU1	99,99	1653	99,99	141
LUU2	99,99	3062	99,99	141
LUU3	99,99	7417	99,99	167
LUU4	99,99	14.234	99,99	221

(*) Todas as Melhores Soluções Publicadas (MSP) são de Hifi (2001).

Tabela 21 - Resultados para o problema *unconstrained two-staged rotated weighted*

Unconstrained Two-staged Rotated Weighted Problem				
ID	**MSP***	**Tempo (seg)**	$A_{PSO+VNS+TF}$	**Tempo (seg)**
UW1	6539	2	6539	33
UW11	17.700	3	17.700	37
UW2	9348	4	9348	45
UW3	6500	4	6500	56
UW4	7748	1	7748	59
UW5	8398	7	8398	45
UW6	6894	12	6894	59
UW7	11.585	16	11.585	62
UW8	8088	22	8088	69
UW9	7527	21	7527	56
UW10	8172	32	8172	74
LUW1	171.079.321	1600	171.079.321	155
LUW2	325.725.070	2911	325.725.070	158
LUW3	433.859.655	7012	433.859.655	320
LUW4	568.436.545	13.636	568.436.545	450

(*) Todas as Melhores Soluções Publicadas (MSP) são de Hifi (2001).

Tabela 22 – Resultados para o problema *constrained two-staged weighted fixed*

ID	**Constrained Two-Staged Weighted Fixed Problem**			
	MSP	**Tempo (seg)**	$A_{PSO+VNS+TF}$	**Tempo (seg)**
HH	10.689	0,01[a]	10.689	6
2	2535	1,5[a]	2535	9
3	1740	0,05[a]	1740	23
A1	1820	<0,01[c]	1820	20
A2	2315	0,11[a]	2315	18
STS2	4620	0,1[c]	4620	34
STS4	9468	0,03[a]	9468	22
CHL1	8360	1,45[a]	8360	31
CHL2	2235	0,07[a]	2235	10
CW1	6402	0,1[c]	6402	27
CW2	5354	0,13[a]	5354	34
Hch12	9630	1,27[a]	9630	33
Hch19	5100	5,15[b]	5100	37

(a) ALGO1_ESGA *algorithm*, Hifi e M'Hallah (2005);
(b) ALGO1_SGA *algorithm*, Hifi e M'Hallah (2005);
(c) Hifi e Roucairol (2001).

Tabela 23 – Resultados para o problema *constrained two-staged unweighted fixed*

ID	Constrained Two-Staged Unweigthed Fixed Problem			
	MSP*	**Tempo (seg)**	**$A_{PSO+VNS+TF}$**	**Tempo (seg)**
2SCUI1	98,03	<1	98,93	43
2SCUI2	97,48	<1	98,17	42
2SCUI3	98,04	<1	98,92	43
2SCUI4	95,59	<1	98,53	41
2SCUI5	98,36	<1	98,36	42
2SCUI6	98,06	<1	98,49	43
2SCUI7	98,31	<1	98,91	43
2SCUI8	98,50	<1	98,9	43
2SCUI9	97,31	<1	98,53	43
2SCUI10	98,11	<1	98,64	42
2SCUI11	96,29	<1	97,98	42
2SCUI12	97,14	<1	97,79	42
2SCUI13	97,96	<1	98,91	43
2SCUI14	96,17	<1	96,98	44
2SCUI15	98,20	<1	98,71	43
2SCUI16	96,71	<1	98,27	43
2SCUI17	94,78	<1	97,14	45
2SCUI18	96,80	<1	97,78	43
2SCUI19	98,32	<1	99,19	42
2SCUI20	94,84	<1	96,61	41
2SCUI21	98,21	<1	98,21	40
2SCUI22	95,68	<1	96,32	45
2SCUI23	96,82	<1	97,73	42
2SCUI24	94,95	<1	95,98	43
2SCUI25	96,02	<1	98,27	44

(*) Todas as Melhores Soluções Publicadas (MSP) são de Cui (2007).

Tabela 24 – Resultados para o problema *constrained two-staged unweighted fixed*

ID	Constrained Two-Staged Unweigthed Fixed Problem			
	MSP*	**Tempo (seg)**	**$A_{PSO+VNS+TF}$**	**Tempo (seg)**
2SCUI26	96,07	<1	98,05	43
2SCUI27	94,34	<1	98,31	43
2SCUI28	96,63	<1	98,51	44
2SCUI29	96,77	<1	98,39	45
2SCUI30	98,27	<1	98,27	41
2SCUI31	94,72	<1	97,78	41
2SCUI32	97,00	<1	97,57	42
2SCUI33	97,97	<1	98,31	42
2SCUI34	97,83	<1	98,09	42
2SCUI35	96,70	<1	97,31	42
2SCUI36	94,35	<1	97,52	43
2SCUI37	94,67	<1	97,36	43
2SCUI38	98,04	<1	98,08	43
2SCUI39	96,75	<1	98,98	44
2SCUI40	97,78	<1	98,21	45
2SCUI41	98,84	<1	98,84	47
2SCUI42	96,92	<1	98,30	43
2SCUI43	96,05	<1	97,41	42
2SCUI44	96,82	<1	97,33	43
2SCUI45	96,84	<1	96,84	43
2SCUI46	95,60	<1	97,63	42
2SCUI47	96,12	<1	97,29	41
2SCUI48	97,86	<1	98,76	45
2SCUI49	98,27	<1	98,27	42
2SCUI50	96,74	<1	98,22	41

(*) Todas as Melhores Soluções Publicadas (MSP) são de Cui (2007)

Tabela 25 – Resultados para o problema *constrained two-staged unweighted rotated*

ID	Constrained Two-Staged Unweigthed Rotated Problem			
	MSP*	**Tempo (seg)**	**$A_{PSO+VNS+TF}$**	**Tempo (seg)**
2SCUI1	98,73	1	99,43	73
2SCUI2	98,21	1	98,55	71
2SCUI3	98,79	1	98,92	71
2SCUI4	98,03	1	98,53	72
2SCUI5	98,36	1	98,61	70
2SCUI6	98,70	1	99,43	70
2SCUI7	98,98	1	99,06	76
2SCUI8	98,67	1	98,93	76
2SCUI9	98,42	1	98,54	77
2SCUI10	99,07	1	99,07	70
2SCUI11	98,44	2	98,44	70
2SCUI12	97,98	1	98,54	71
2SCUI13	98,78	1	98,91	70
2SCUI14	98,14	1	98,14	70
2SCUI15	98,93	1	99,15	70
2SCUI16	98,75	1	98,75	72
2SCUI17	98,33	1	98,33	75
2SCUI18	98,44	1	98,44	76
2SCUI19	98,86	1	99,19	76
2SCUI20	96,96	1	97,17	70
2SCUI21	98,78	1	98,78	72
2SCUI22	97,40	1	97,40	71
2SCUI23	98,43	1	98,81	76
2SCUI24	97,45	2	97,45	70
2SCUI25	97,77	1	98,49	75

(*) Todas as Melhores Soluções Publicadas (MSP) são de Cui (2007)

Tabela 26 - Resultados para o problema *constrained two-staged unweighted rotated*

ID	Constrained Two-Staged Unweigthed Rotated Problem			
	MSP*	**Tempo (seg)**	**$A_{PSO+VNS+TF}$**	**Tempo (seg)**
2SCUI26	98,18	1	98,32	74
2SCUI27	97,23	1	98,31	73
2SCUI28	98,90	1	98,90	69
2SCUI29	99,45	1	99,45	67
2SCUI30	98,27	1	98,36	74
2SCUI31	98,07	1	98,42	75
2SCUI32	98,13	1	98,67	71
2SCUI33	99,30	1	99,30	72
2SCUI34	99,20	1	99,20	71
2SCUI35	98,39	1	98,39	71
2SCUI36	97,92	1	98,37	72
2SCUI37	98,26	1	98,26	76
2SCUI38	98,89	1	98,98	73
2SCUI39	98,76	1	98,98	74
2SCUI40	98,28	1	98,28	72
2SCUI41	99,15	1	99,40	71
2SCUI42	99,17	1	99,17	72
2SCUI43	98,34	1	98,34	71
2SCUI44	98,13	1	98,20	71
2SCUI45	98,73	1	98,73	72
2SCUI46	97,84	1	98,13	73
2SCUI47	97,68	2	98,17	71
2SCUI48	98,57	1	98,76	76
2SCUI49	98,85	2	98,85	69
2SCUI50	97,99	1	98,43	68

(*) Todas as Melhores Soluções Publicadas (MSP) são de Cui (2007)

Tabela 27 - Resultados para o problema *unconstrained three-staged fixed unweighted*

Unconstrained Three-staged Fixed Unweigthed Problem				
ID	**MSP***	**Tempo (seg)**	$A_{PSO+VNS+TF}$	**Tempo (seg)**
UU1	96,50	380	96,50	25
UU2	99,21	1349	99,21	30
UU3	97,52	3148	**96,82**	26
UU4	98,19	3787	**97,39**	36
UU5	99,15	10.906	99,15	46
UU6	98,79	10.843	98,79	35
UU7	98,84	11.055	**98,61**	50
UU8	98,98	10.968	98,98	54
UU9	99,20	10.960	99,20	59
UU10	99,01	11.320	**98,30**	57
UU11	99,76	15.353	**99,73**	24
LUU1	N/A	N/A	99,95	76
LUU2	N/A	N/A	99,96	75
LUU3	N/A	N/A	99,98	89
LUU4	N/A	N/A	98,42	130

(*) Todas as Melhores Soluções Publicadas (MSP) são de Hifi (2001). O marcador N/A significa que não estão disponíveis.

Tabela 28 - Resultados para o problema *unconstrained three-staged fixed weighted*

Unconstrained Three-staged Fixed Weighted Problem				
ID	**MSP***	**Tempo (seg)**	$A_{PSO+VNS+TF}$	**Tempo (seg)**
UW1	6036	390	6036	17
UW11	15.747	798	15.747	19
UW2	8468	894	8468	23
UW3	6226	946	6226	28
UW4	8326	4284	8326	28
UW5	7780	5592	7780	22
UW6	6615	10.977	6615	29
UW7	10.464	11.101	10.464	31
UW8	7692	11.252	7692	35
UW9	7038	11.058	7038	28
UW10	7507	11.732	7507	37
LUW1	N/A	N/A	158.363.374	76
LUW2	N/A	N/A	257.372.669	77
LUW3	N/A	N/A	261.252.077	160
LUW4	N/A	N/A	473.754.165	226

(*) Todas as Melhores Soluções Publicadas (MSP) são de Hifi (2001). O marcador N/A significa que não estão disponíveis.

Tabela 29 - Resultados para o problema *unconstrained three-staged rotated unweighted*

Unconstrained Three-staged Rotated Unweighted Problem				
ID	**MSP***	**Tempo (seg)**	$A_{PSO+VNS+TF}$	**Tempo (seg)**
UU1	98,42	451	98,42	54
UU2	99,21	1579	99,21	61
UU3	99,03	3480	99,03	52
UU4	99,05	4415	98,97	73
UU5	99,64	11.152	99,64	99
UU6	98,81	11.063	98,81	79
UU7	99,46	11.742	98,67	105
UU8	99,37	11.244	98,98	111
UU9	99,36	11.540	99,2	126
UU10	99,22	13.039	99,16	112
UU11	99,88	19.793	99,86	61
LU1	N/A	N/A	99,99	151
LU2	N/A	N/A	99,99	149
LU3	N/A	N/A	99,99	182
LU4	N/A	N/A	99,98	241

(*) Todas as Melhores Soluções Publicadas (MSP) são de Hifi (2001). O marcador N/A significa que não estão disponíveis.

Tabela 30 - Resultados para o problema *unconstrained three-staged rotated weighted*

Unconstrained Three-staged Rotated Weighted Problem				
ID	**MSP***	**Tempo (seg)**	$A_{PSO+VNS+TF}$	**Tempo (seg)**
UW1	6696	1024	6696	44
UW11	18.200	2269	18.200	50
UW2	9732	2451	9732	48
UW3	7188	2632	7188	60
UW4	8452	5174	8452	82
UW5	8398	6262	8398	61
UW6	6937	11.400	6937	88
UW7	11.585	11.900	11.585	92
UW8	8088	12.016	8088	104
UW9	7527	12.375	7527	92
UW10	8172	12.086	8172	82
LUW1	N/A	N/A	171.836.653	159
LUW2	N/A	N/A	326.399.541	162
LUW3	N/A	N/A	433.882.185	331
LUW4	N/A	N/A	568.690.356	461

(*) Todas as Melhores Soluções Publicadas (MSP) são de Hifi (2001). O marcador N/A significa que não estão disponíveis.

Tabela 31 - Resultados para o problema *constrained three-staged weighted fixed*

ID	Constrained Three-Staged Weighted Fixed Problem			
	MSP	**Tempo (seg)**	**$A_{PSO+VNS+TF}$**	**Tempo (seg)**
3SCUI1	631.863	6[a]	**631.852**	51
3SCUI2	4.544.395	68[a]	4.544.395	65
3SCUI3	1.833.000	10[c]	1.833.000	83
3SCUI4	4.339.083	9[c]	4.339.083	64
3SCUI5	2.676.651	4[b]	2.676.651	53
3SCUI6	4.245.663	4[b]	4.245.663	53
3SCUI7	4.305.010	15[c]	4.305.010	87
3SCUI8	3.610.441	16[a]	**3.608.015**	76
3SCUI9	7.446.330	9[b]	7.446.330	60
3SCUI10	6.038.211	3600[a]	6.038.211	74
3SCUI11	1.168.284	4[a]	**1.162.898**	56
3SCUI12	2.567.171	4[c]	2.567.171	69
3SCUI13	2.795.973	4[b]	2.795.973	52
3SCUI14	3.152.403	3[c]	3.152.403	53
3SCUI15	5.018.587	12[b]	5.018.587	87
3SCUI16	2.157.408	9[b]	2.157.408	81
3SCUI17	6.881.547	18[a]	**6.854.864**	56
3SCUI18	7.548.827	114[a]	7.548.827	54
3SCUI19	5.704.261	11[b]	5.704.261	70
3SCUI20	5.726.840	1293[a]	**5.676.748**	84

(a) VEA, Cui (2008);

(b) VHA1, Cui (2008);

(c) VHA2, Cui (2008).

Tabela 32 - Resultados para o problema *constrained three-staged unweighted fixed*

ID	Constrained Three-Staged Unweigthed Fixed Problem			
	MSP	**Tempo (seg)**	**$A_{PSO+VNS+TF}$**	**Tempo (seg)**
3SCUI1	99,78	1[b]	99,78	95
3SCUI2	99,73	7[b]	99,73	94
3SCUI3	99,83	1[c]	99,83	147
3SCUI4	99,84	10[b]	99,84	101
3SCUI5	99,77	5[c]	99,77	82
3SCUI6	99,69	5[b]	99,69	87
3SCUI7	99,79	15[b]	99,79	153
3SCUI8	99,82	8[b]	99,82	134
3SCUI9	99,75	10[b]	99,75	111
3SCUI10	99,77	13[b]	99,77	124
3SCUI11	99,67	5[a]	**99,65**	79
3SCUI12	99,78	4[a]	99,78	90
3SCUI13	99,72	4[b]	99,72	85
3SCUI14	99,61	3[c]	99,61	90
3SCUI15	99,84	13[b]	99,84	145
3SCUI16	99,82	10[b]	99,82	149
3SCUI17	99,72	8[b]	99,72	109
3SCUI18	99,59	8[b]	99,59	98
3SCUI19	99,77	12[b]	99,77	134
3SCUI20	99,75	14[b]	99,75	150

(a) VEA *algorithm*, Cui (2008);
(b) VHA1algortihm, Cui (2008) ;
(c) VHA2 *algorithm*, Cui (2008) .

Tabela 33 - Resultados para o problema non-*guillotine first-order unconstrained weighted fixed*

ID	Non-guillotine First Order Unconstrained Weighted Fixed Problem			
	MSP*	**Tempo (seg)**	**$A_{PSO+VNS+TF}$**	**Tempo (seg)**
UW1	6036	N/A	6036	92
UW2	8720	N/A	8720	135
UW3	6652	N/A	6652	106
UW4	8326	N/A	8326	132
UW5	7780	N/A	7780	109
UW6	6803	N/A	**6615**	141
UW7	10.464	N/A	10.464	131
UW8	7692	N/A	7692	155
UW9	7128	N/A	**7038**	131
UW10	7507	N/A	7507	169
UW11	16.400	N/A	**15.920**	174

(*) Todas as Melhores Soluções Publicadas (MSP) são de Birgin et al. (2012). O marcador N/A significa que não estão disponíveis.

Tabela 34 - Resultados para o problema non-*guillotine first order unconstrained unweighted fixed*

ID	Non-guillotine First Order Unconstrained Unweighted Fixed Problem			
	MSP*	**Tempo (seg)**	**$A_{PSO+VNS+TF}$**	**Tempo (seg)**
UU1	245.205	N/A	245.205	103
UU2	595.288	N/A	595.288	136
UU3	1088.154	N/A	**1081.136**	85
UU4	1191.071	N/A	**1178.295**	104
UU5	1870.038	N/A	1870.038	140
UU6	2950.760	N/A	2950.760	128
UU7	2943.852	N/A	**2932.092**	164
UU8	3969.784	N/A	3969.784	153
UU9	6100.692	N/A	6100.692	144
UU10	11.955.852	N/A	11.955.852	183
UU11	13.157.811	N/A	**13.141.175**	137

(*) Todas as Melhores Soluções Publicadas (MSP) são de Birgin et al. (2012). O marcador N/A significa que não estão disponíveis.

Tabela 35 - Resultados para o problema non-*guillotine superior order unconstrained weighted fixed*

ID	Non-guillotine Superior Order Unconstrained weighted Fixed Problem			
	MSP*	**Tempo (seg)**	**$A_{PSO+VNS+TF}$**	**Tempo (seg)**
ATP20	5545.818	N/A	5545.818	155
ATP21	3491.545	N/A	**3484.406**	212
ATP22	4153.426	N/A	**4145.317**	188
ATP23	3546.535	N/A	**3480.665**	244
ATP24	3948.037	N/A	**3923.776**	188
ATP25	3507.615	N/A	3507.615	245
ATP26	2683.689	N/A	2683.689	247
ATP27	2438.174	N/A	2438.174	280
ATP28	4065.011	N/A	**3969.356**	225
ATP29	3652.858	N/A	3652.858	276

(*) Todas as Melhores Soluções Publicadas (MSP) são de Alvarez-Valdes et al. (2007). O marcador N/A significa que não estão disponíveis.

Tabela 36 - Resultados para o problema non-*guillotine superior order unconstrained unweighted fixed*

ID	Non-guillotine Superior Order Unconstrained Unweighted Fixed Problem			
	MSP*	**Tempo (seg)**	**$A_{PSO+VNS+TF}$**	**Tempo (seg)**
ATP10	3589.703	N/A	**3585.612**	132
ATP11	4188.915	N/A	4188.915	133
ATP12	5156.065	N/A	**5148.302**	170
ATP13	3498.302	N/A	3498.302	165
ATP14	4463.550	N/A	4463.550	102
ATP15	6047.188	N/A	6047.188	148
ATP16	7566.719	N/A	7566.719	156
ATP17	4535.302	N/A	4535.302	179
ATP18	5825.956	N/A	5825.956	121
ATP19	6826.674	N/A	6826.674	213

(*) Todas as melhores soluções publicadas (MSP) são de Alvarez-Valdes et al. (2007). O marcador N/A significa que não estão disponíveis.

Tabela 37 - Resultados para o problema non-*guillotine superior order constrained weighted fixed*

ID	Non-guillotine Superior Order Constrained Weighted Fixed Problem			
	MSP*	**Tempo (seg)**	**$A_{PSO+VNS+TF}$**	**Tempo (seg)**
Beasley1	164	N/A	164	106
Beasley2	230	N/A	**225**	127
Beasley3	247	N/A	**220**	146
Beasley4	268	N/A	268	190
Beasley5	358	N/A	**301**	98
Beasley6	289	N/A	**229**	119
Beasley7	430	N/A	430	123
Beasley8	834	N/A	**820**	201
Beasley9	924	N/A	924	107
Beasley10	1452	N/A	1452	116
Beasley11	1688	N/A	1688	159
Beasley12	1865	N/A	1801	185

(*) Todas as Melhores Soluções Publicadas (MSP) são de Egeblad e Pisinger, (2009). O marcador N/A significa que não estão disponíveis.

Tabela 38 - Resultados para o problema non-*guillotine superior order constrained unweighted fixed*

ID	Non-guillotine Superior Order Constrained Unweighted Fixed Problem			
	MSP*	**Tempo (seg)**	**$A_{PSO+VNS+T}$**	**Tempo (seg)**
Lai&Chan1	80.000	N/A	**80.000**	89
Lai&Chan 2	79.000	N/A	76.000	96
Lai&Chan 3	160.000	N/A	154.600	99
Jakobs1	5600	N/A	5447	85
Jakobs2	5600	N/A	5455	72
Jakobs3	5400	N/A	5328	136
Jakobs4	4050	N/A	3978	114
Jakobs5	2925	N/A	2871	186
Leung1	16.500	N/A	16.196	148
Leung2	19.200	N/A	18.628	140

(*) Todas as melhores soluções publicadas (MSP) são de Alvarez-Valdes et al. (2007). O marcador N/A significa que não estão disponíveis.

Tabela 39 - Resultados para o problema *guillotine non-staged unconstrained weighted rotated*

ID	Unconstrained Non-staged Weigthed Rotated Problem	
	$A_{PSO+VNS+TF}$	Tiempo (seg)
UW1	6696	56
UW11	18.200	63
UW2	9732	54
UW3	7188	76
UW4	8452	103
UW5	8398	82
UW6	6937	110
UW7	11.585	115
UW8	8088	130
UW9	7527	116
UW10	8172	63
LUW1	171.836.653	156
LUW2	326.399.541	154
LUW3	433.882.185	188
LUW4	568.690.356	249

Tabela 40 - Resultados para o problema *guillotine non-staged constrained weighted rotated*

ID	Constrained Non-staged Weighted Rotated Problem	
	$A_{PSO+VNS+TF}$	Tiempo (seg)
CW1	6764	124
CW2	5540	126
CW3	5689	202
CW4	7466	186
CW5	11.659	202
CW6	12.959	320
CW7	10.880	310
CW8	4736	366
CW9	11.479	280
CW10	6705	280
CW11	6604	250
TH1	4630	168
TH2	9683	104
CHW1	2858	52
CHW2	1900	126

Tabela 41 - Resultados para o problema *guillotine non-staged constrained unweighted rotated*

ID	Constrained Non-staged Unweighted Rotated Problem	
	$A_{PSO+VNS+TF}$	Tiempo (seg)
CU1	98,64	203
CU2	99,43	210
CU3	99,84	403
CU4	99,22	365
CU5	98,75	387
CU6	99,65	540
CU7	97,5	570
CU8	99,2	611
CU9	98,32	540
CU10	98,94	540
CU11	99,31	470

Tabela 42 - Resultados para o problema *guilltoine two-staged unconstrained fixed weighted e unweighted*

Unconstrained Two-staged Fixed Problem					
Unweighted			Weighted		
ID	$A_{PSO+VNS+TF}$	**Tiempo (seg)**	**ID**	$A_{PSO+VNS+TF}$	**Tiempo (seg)**
UU1	96,50	25	**UW1**	6036	17
UU2	99,21	28	**UW11**	15.747	19
UU3	97,52	26	**UW2**	8468	25
UU4	98,19	34	**UW3**	6226	29
UU5	99,15	47	**UW4**	7748	32
UU6	98,79	36	**UW5**	7780	24
UU7	98,84	48	**UW6**	6615	31
UU8	98,98	51	**UW7**	10.464	30
UU9	99,20	58	**UW8**	7692	34
UU10	99,01	53	**UW9**	7038	29
UU11	99,49	27	**UW10**	7507	38
LUU1	99,98	70	**LUW1**	158.363.374	80
LUU2	99,99	70	**LUW2**	257.372.669	83
LUU3	99,99	80	**LUW3**	261.252.077	180
LUU4	99,98	112	**LUW4**	473.754.165	230

Tabela 43 - Resultados para o problema *guillotine two-staged constrained weighted rotated*

ID	Constrained Two-Staged Weighted Rotated Problem	
	$A_{PSO+VNS+TF}$	Tiempo (seg)
HH	10.710	10
2	2539	21
3	1746	42
A1	1835	37
A2	2327	41
STS2	4646	56
STS4	9468	39
CHL1	8382	60
CHL2	2235	17
CW1	6425	50
CW2	5356	69
Hch12	9684	64
Hch19	5104	61

Tabela 44 - Resultados para o problema *guillotine three-staged constrained weighted e unweigthed rotated*

ID	Constrained Three-Staged Rotated Problem			
	Unweighted		Weighted	
	$A_{PSO+VNS+TF}$	Tiempo (seg)	$A_{PSO+VNS+TF}$	Tiempo (seg)
3SCUI1	99,98	180	632.402	94
3SCUI2	99,73	178	4545.920	120
3SCUI3	99,93	276	1833.944	153
3SCUI4	99,98	198	4339.962	118
3SCUI5	99,99	190	2678.314	98
3SCUI6	99,83	197	4249.533	99
3SCUI7	99,79	300	4306.566	160
3SCUI8	99,82	256	3610.791	140
3SCUI9	99,83	223	7447.743	111
3SCUI10	99,81	245	6040.071	136
3SCUI11	99,78	163	1169.181	103
3SCUI12	99,99	187	2567.626	127
3SCUI13	99,78	176	2796.196	96
3SCUI14	99,76	190	3153.305	98
3SCUI15	99,84	281	5021.789	160
3SCUI16	99,82	272	2158.001	149
3SCUI17	99,79	209	6885.015	103
3SCUI18	99,85	203	7552.210	99
3SCUI19	99,77	265	5709.350	129
3SCUI20	99,75	150	5731.970	155

Tabela 45 - Resultados para o problema *non-guillotine first order unconstrained unweighted*

Unconstrained Non-guillotine First Order Unweighted Fixed Problem		
	$A_{PSO+VNS+TF}$	**Tiempo (seg)**
GCUT1	56.460	60
GCUT2	61.146	73
GCUT3	61.036	87
GCUT4	61.698	111
GCUT5	246.000	71
GCUT6	238.998	59
GCUT7	242.567	67
GCUT8	247.815	88
GCUT9	971.100	70
GCUT10	982.025	54
GCUT11	980.096	51
GCUT12	979.986	104
GCUT13	8997.780	79

Tabela 46 - Resultados para o problema *non-guillotine superior order unconstrained weighted rotated*

Non-guillotine Superior Order Unconstrained weighted Fixed Problem		
	$A_{PSO+VNS+TF}$	**Tiempo (seg)**
ATP20	5545.818	200
ATP21	3484.406	409
ATP22	4145.317	287
ATP23	3480.665	426
ATP24	3923.776	276
ATP25	3507.615	337
ATP26	2683.689	354
ATP27	2438.174	532
ATP28	3969.356	399
ATP29	3652.858	497

Tabela 47 - Resultados para o problema *non-guillotine superior order unconstrained unweighted rotated*

Non-guillotine Superior Order Unconstrained Unweighted Fixed Problem		
	$A_{PSO+VNS+TF}$	**Tiempo (seg)**
ATP10	3585.612	107
ATP11	4188.915	95
ATP12	5148.302	162
ATP13	3498.302	297
ATP14	4463.550	227
ATP15	6047.188	228
ATP16	7566.719	241
ATP17	4535.302	116
ATP18	5825.956	218
ATP19	6826.674	162