LEANDRO MACIEL TURI

**Métodos heurísticos para resolução de problemas de empacotamento
unidimensional**

São Paulo
(2018)

LEANDRO MACIEL TURI

**Métodos heurísticos para resolução de problemas de empacotamento
unidimensional**

Dissertação apresentada à Escola
Politécnica da Universidade de São Paulo
para obtenção do título de Mestre em
Ciências

Orientador: Prof. Dr. André Bergsten
Mendes

São Paulo
(2018)

LEANDRO MACIEL TURI

**Métodos heurísticos para resolução de problemas de empacotamento
unidimensional**

Dissertação apresentada à Escola
Politécnica da Universidade de São Paulo
para obtenção do título de Mestre em
Ciências

Área de Concentração: Engenharia de
Sistemas Logísticos

Orientador: Prof. Dr. André Bergsten
Mendes

São Paulo
(2018)

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

São Paulo, _____ de _____ de _____

Assinatura do autor: _____

Assinatura do orientador: _____

Catálogo-na-publicação

Turi, Leandro Maciel

Métodos heurísticos para resolução de problemas de empacotamento unidimensional / L. M. Turi – versão corr. -- São Paulo, 2018.
118 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Naval e Oceânica.

1.Problema de empacotamento unidimensional 2.Métodos heurísticos
3.Limitantes inferiores 4.Simulated Annealing I.Universidade de São Paulo.
Escola Politécnica. Departamento de Engenharia Naval e Oceânica II.t.

À minha filha Julia

AGRADECIMENTOS

Ao meu orientador professor Dr. André Bergsten Mendes, pelos ensinamentos passados e orientações valiosas que contribuíram muito para a minha pesquisa.

À minha esposa Bruna por todo o apoio recebido e pela paciência comigo.

À minha filha Julia por toda a paciência e compreensão ao meu trabalho e às minhas ausências por ele provocadas.

Aos meus pais André e Terezinha que sempre me apoiaram e incentivaram a seguir em frente para realizar meus sonhos.

À minha irmã Débora, meu cunhado André e minha sobrinha Lívia, minha família, a base sólida que me faz seguir em frente.

Ao professor Dr. Miguel Santoro pelas valiosas contribuições à pesquisa.

Ao professor Dr. Cláudio Barbieri da Cunha pelos ensinamentos prestados e grande apoio nos artigos científicos publicados.

A Metalúrgica Albras Ltda por autorizar minhas faltas e saídas para as aulas e em particular ao meu gerente Paulo Franco por todo o apoio, sem o qual não seria possível fazer esta pesquisa.

Escolhe um trabalho de que gostes, e não
terás que trabalhar nem um dia na tua vida.
Confúcio

RESUMO

Os problemas de corte e empacotamento são muito comuns nas indústrias e na logística. Dado um conjunto de N itens com diferentes pesos e um conjunto de M contentores com capacidade C , o problema de empacotamento unidimensional consiste em determinar o menor número de contentores a serem utilizados para alocar todos os itens respeitando a restrição de capacidade dos contentores. Nesse estudo pretende-se resolver o problema com instâncias *benchmark* da literatura, por meio de sessenta heurísticas diferentes, que são comparadas a quatro limitantes inferiores propostos na literatura com o intuito de avaliar a qualidade da solução heurística. Quatro limitantes inferiores e dez heurísticas construtivas diferentes foram programados em C++ num mesmo ambiente computacional, permitindo sua comparação tanto em termos de qualidade das soluções, quanto em termos dos tempos de processamento. Uma heurística simples de troca de itens entre contentores chamada Diferença-de-Quadrados foi proposta para melhorar as soluções iniciais do problema. A metaheurística *simulated annealing* foi acionada para melhorar a solução inicial quando o limitante inferior não foi atingido. Os parâmetros dos *simulated annealing* foram determinados com os dados das instâncias de forma diferente da utilizada na literatura. As combinações entre as dez soluções iniciais, a heurística Diferença-de-Quadrados e o *simulated annealing* geraram um conjunto de sessenta heurísticas diferentes. Os resultados mostraram que o algoritmo proposto é eficiente para resolver o problema com tempos de processamento adequados a tomada de decisão.

Palavras-Chave: Problema de empacotamento unidimensional, Métodos heurísticos, Limitantes inferiores, *Simulated Annealing*.

ABSTRACT

Cutting and packing problems are very common in industries and logistics. Given a set of N items with different weights and a set of M bins with full capacity C , the one-dimensional bin packing problem consists of determining the smallest number of bins capable to allocate all items respecting the capacity constraint of the bins. that impose that the sum of the weights of the items allocated to the bin is less than or equal to their capacity. In this study we intend to solve the problem with benchmark instances of the literature, by means of sixty different heuristics, which are compared to four lower bounds proposed in the literature in order to evaluate the quality of the heuristic solution. Four lower bounds and ten different constructive heuristics were programmed in C++ in the same computational environment, allowing their comparison both in terms of the quality of the solutions and in terms of processing times. A simple heuristic of item exchange between bins called Difference-of-Squares was proposed to improve the initial solutions of the problem. The simulated annealing metaheuristic was triggered to improve the initial solution when the lower bounds was not reached. The parameters of the simulated annealing were determined with the data of the instances differently from that used in the literature. The combinations of the ten initial solutions, the Difference-of-Squares heuristic and the simulated annealing generated a set of sixty different heuristics. The results showed that the proposed algorithm is efficient to solve the problem with adequate processing times for decision making.

Keywords: Bin packing problem, Heuristic method, Lower bounds, Simulated Annealing.

LISTA DE ILUSTRAÇÕES

Figura 1 – Tipos básicos de problemas de corte e empacotamento	23
Figura 2 – Pseudocódigo do cálculo do limitante inferior L2 de Martello e Toth	38
Figura 3 – Pseudocódigo do procedimento de redução MTRP de Martello e Toth ...	39
Figura 4 – Pseudocódigo do cálculo do limitante inferior L3 de Martello e Toth	40
Figura 5 – Pseudocódigo do cálculo do limitante inferior LBPP	41
Figura 6 – Pseudocódigo da heurística Sum-of-Squares – SSD	43
Figura 7 – Pseudocódigo da heurística <i>Minimal Bin Slack</i> – MBS de Gupta e Ho	44
Figura 8 – Pseudocódigo da heurística <i>First Fit Decreasing</i> – FFD	47
Figura 9 – Pseudocódigo da heurística Last Fit Decreasing – LFD	48
Figura 10 – Pseudocódigo da heurística <i>Best Fit Decreasing</i> – BFD	49
Figura 11 – Pseudocódigo da heurística <i>Worst Fit Decreasing</i> – WFD	50
Figura 12 – Pseudocódigo da heurística <i>Next Fit Decreasing</i> – NFD	51
Figura 13 – Pseudocódigo da heurística <i>Worst Solution</i> – WS	51
Figura 14 – Exemplo de movimento de inserção de um item γ em um contentor β ..	56
Figura 15 – Exemplo de movimento de troca de itens entre contentores	57
Figura 16 – Exemplo de movimento de troca do tipo um item por dois itens	57
Figura 17 – Exemplo de movimento de troca do tipo dois itens por dois itens	58
Figura 18 – Pseudocódigo da heurística diferença de quadrados	60
Figura 19 – Instância utilizada no exemplo da heurística diferença de quadrados ...	61
Figura 20 – Exemplo de solução inicial gerada pela heurística FFD	62
Figura 21 – Exemplo de movimento de inserção inviável.	63
Figura 22 – Exemplo de movimento de troca com redução na função objetivo	64
Figura 23 – Troca entre contentores do item 1 com o item 12	64
Figura 24 – Troca entre contentores dos itens 2 e 3 com melhoria na função objetivo	65
Figura 25 – Movimento de inserção viável do item 19 no contentor 4	65
Figura 26 – Redução no número de contentores com a heurística diferença de quadrados	66
Figura 27 – Pseudocódigo do <i>Simulated Annealing</i> - SA	72
Figura 28 – Fluxograma do programa para resolução dos BPP	74
Figura 29 – Tempo total de processamento (segundos) por heurística inicial com número máximo de iterações sem melhoria igual a 50	90
Figura 30 – Percentual do tempo total de processamento por tipo de solução inicial e parte do algoritmo com número máximo de iterações sem melhorias igual a 50	91
Figura 31 – Tempo total de processamento (em segundos) por parte do algoritmo com número máximo de iterações sem melhoria igual a 100	93
Figura 32 – Percentual do tempo total de processamento por parte do algoritmo com número máximo de iterações sem melhorias igual a 100	93
Figura 33 – Tempo total de processamento (em segundos) por parte do algoritmo com número máximo de iterações sem melhoria igual a 200	95
Figura 34 – Percentual do tempo total de processamento por parte do algoritmo com número máximo de iterações sem melhorias igual a 200	96
Figura 35 – Probabilidade de aceitar soluções piores com $\sigma = 0,95$ e parâmetros da primeira estimativa	107
Figura 36 – Probabilidade de aceitar soluções piores com $\sigma = 0,96$ e parâmetros da primeira estimativa	107
Figura 37 – Probabilidade de aceitar soluções piores com $\sigma = 0,97$ e parâmetros da primeira estimativa	108

Figura 38 – Probabilidade de aceitar soluções piores com $\sigma = 0,98$ e parâmetros da primeira estimativa	108
Figura 39 – Probabilidade de aceitar soluções piores com $\sigma = 0,99$ e parâmetros da primeira estimativa	109
Figura 40 – Número de soluções avaliadas e tempo de processamento em função de σ e parâmetros da primeira estimativa	110
Figura 41 – Probabilidade de aceitar soluções piores com $\sigma = 0,95$ e parâmetros da segunda estimativa	111
Figura 42 – Probabilidade de aceitar soluções piores com $\sigma = 0,96$ e parâmetros da segunda estimativa	111
Figura 43 – Probabilidade de aceitar soluções piores com $\sigma = 0,97$ e parâmetros da segunda estimativa	112
Figura 44 – Probabilidade de aceitar soluções piores com $\sigma = 0,98$ e parâmetros da segunda estimativa	112
Figura 45 – Probabilidade de aceitar soluções piores com $\sigma = 0,99$ e parâmetros da segunda estimativa	113
Figura 46 – Número de soluções avaliadas e tempo de processamento em função de σ e parâmetros da segunda estimativa	113

LISTA DE TABELAS

Tabela 1 – Resumo dos dados das principais instâncias da literatura	78
Tabela 2 – Percentual de soluções ótimas e tempos de processamento (horas) versus número de itens na instância	80
Tabela 3 – Diferença média e máxima entre o limitante superior e o inferior versus número de itens na instância.....	80
Tabela 4 – Resultados dos experimentos computacionais	81
Tabela 5 – Resultados dos experimentos computacionais (continuação).....	82
Tabela 6 – Resultados das heurísticas com solução inicial MBS.....	83
Tabela 7 – Resultados das heurísticas com solução inicial MBS'	84
Tabela 8 – Resultados das heurísticas com solução inicial MBSR	85
Tabela 9 – Resultados das heurísticas com solução inicial FFD.....	86
Tabela 10 – Resultados das heurísticas com solução inicial BFD	86
Tabela 11 – Resultados das heurísticas com solução inicial WS.....	86
Tabela 12 – Resultados das heurísticas com solução inicial SSD	87
Tabela 13 – Resultados das heurísticas com solução inicial LFD.....	88
Tabela 14 – Resultados das heurísticas com solução inicial WFD	88
Tabela 15 – Resultados das heurísticas com solução inicial NFD	88
Tabela 16 – Tempo total de processamento (segundos) por parte do algoritmo com número máximo de iterações sem melhorias igual a 50	89
Tabela 17 – Percentual do tempo de processamento por tipo de solução inicial e parte do algoritmo com número máximo de iterações igual a 50	92
Tabela 18 – Tempo total de processamento (segundos) por parte do algoritmo com número máximo de iterações sem melhorias igual a 100	92
Tabela 19 – Percentual do tempo de processamento por tipo de solução inicial e parte do algoritmo com número máximo de iterações igual a 100	94
Tabela 20 – Tempo total de processamento (segundos) por parte do algoritmo com número máximo de iterações sem melhorias igual a 200	94
Tabela 21 – Percentual do tempo de processamento por tipo de solução inicial e parte do algoritmo com número máximo de iterações igual a 200	95
Tabela 22 – Resultados do teste de robustez das heurísticas do tipo HEUR+DS+SA	97
Tabela 23 – Tempo médio de processamento por rodada para duas instâncias de 500 itens.....	98

LISTA DE ABREVIATURAS E SIGLAS

BF	<i>Best Fit procedure</i>
BFD	<i>Best Fit Decreasing procedure</i>
BISON	<i>Bin Packing Solution procedure</i>
BPP	<i>Bin Packing Problem</i>
BPPC	<i>Bin Packing Problem with Conflicts</i>
CSP	<i>Cutting Stock Problem</i>
DS	<i>Diferença de Quadrados – Difference-of-Squares</i>
FF	<i>First Fit procedure</i>
FFD	<i>First Fit Decreasing procedure</i>
GRASP	<i>Greedy Randomized Adaptive Search Procedure</i>
HEUR	<i>Heurística construtiva</i>
HGGA	<i>Hybrid Grouping Genetic Algorithm for Bin Packing</i>
HI_BP	<i>Hybrid Improvement procedure for the bin packing problem</i>
LFD	<i>Last Fit Decreasing procedure</i>
LI	<i>Limitante inferior</i>
LS	<i>Limitante superior</i>
MBS	<i>Minimal Bin Slack procedure</i>
MBS'	<i>Minimal Bin Slack' procedure</i>
MBSR	<i>Relaxed Minimal Bin Slack procedure</i>
MTP	<i>Branch-and-Bound procedure de Martello & Toth</i>
MTPCS	<i>Procedure MTP com Limitante Inferior de Cutting Stock Problem</i>
MTRP	<i>Martello & Toth Reduction Procedure</i>
NF	<i>Next Fit procedure</i>
NFD	<i>Next Fit Decreasing procedure</i>
PCE	<i>Problemas de corte e empacotamento</i>
PMBS	<i>Perturbation Minimal Bin Slack procedure</i>
SA	<i>Simulated Annealing</i>
SALBP	<i>Simple Assembly Line Balancing Problem</i>
SHP	<i>Sequential heuristic procedure</i>
SS	<i>Sum-of-Squares procedure</i>
SSD	<i>Sum-of-Squares Decreasing procedure</i>

VNS	<i>Variable Neighborhood Search</i>
WA	<i>Weighted Annealing</i>
WABP	<i>Weighted Annealing Bin Packing procedure</i>
WS	<i>Worst Solution procedure</i>
WFD	<i>Worst Fit Decreasing procedure</i>

Sumário	
RESUMO.....	8
LISTA DE ILUSTRAÇÕES	10
LISTA DE TABELAS	12
LISTA DE ABREVIATURAS E SIGLAS	13
1 INTRODUÇÃO	13
1.1 DEFINIÇÃO DO PROBLEMA E MOTIVAÇÃO	15
1.2 OBJETIVOS PRINCIPAIS	15
1.3 OBJETIVOS SECUNDÁRIOS	16
1.4 MÉTODO ESCOLHIDO	16
1.5 ESTRUTURA DA DISSERTAÇÃO	16
2 REVISÃO DA LITERATURA	18
2.1 TIPOLOGIA DOS PROBLEMAS DE CORTE E EMPACOTAMENTO.....	18
2.2 ABORDAGENS EXATAS E HEURÍSTICAS PARA OS PROBLEMAS DE CORTE E EMPACOTAMENTO.....	24
2.3 LIMITANTES INFERIORES.....	35
2.3.1 O limitante inferior de Johnson	35
2.3.2 O limitante inferior L_1 do contínuo	36
2.3.3 O limitante inferior L_2 de Martello e Toth (1990).....	36
2.3.4 O procedimento de redução de Martello e Toth	39
2.3.5 O limitante inferior L_3 de Martello e Toth	40
2.3.6 Melhor limitante inferior utilizado	41
2.4 LIMITANTES SUPERIORES	42
2.4.1 Heurística <i>Sum-of-Squares</i> - SSD.....	42
2.4.2 Heurística <i>Minimal Bin Slack</i> – MBS.....	44
2.4.3 Heurística <i>Minimal Bin Slack</i> modificada – MBS'	45
2.4.4 Heurística <i>Minimal Bin Slack Relaxed</i> – MBSR	46
2.4.5 Heurística <i>First Fit Decreasing</i> - FFD	46
2.4.6 Heurística <i>Last Fit Decreasing</i> - LFD	47
2.4.7 Heurística <i>Best Fit Decreasing</i> - BFD	48
2.4.8 Heurística <i>Worst Fit Decreasing</i> - WFD	49
2.4.9 Heurística <i>Next Fit Decreasing</i> - NFD	50
2.4.10 Heurística <i>Worst Solution</i> - WS.....	51
3 MODELAGEM MATEMÁTICA E MÉTODOS DE SOLUÇÃO	52
3.1 MODELAGEM MATEMÁTICA	52
3.1.1 Modelo matemático do Problema de Empacotamento Unidimensional	52
3.1.2 Reformulação da Função Objetivo	53
3.2 MÉTODOS DE SOLUÇÃO	58
3.2.1 Heurística Diferença de Quadrados.....	59
3.2.2 Exemplo da heurística Diferença de Quadrados	61
3.2.3 Metaheurística <i>simulated annealing</i>	66
4 TESTES COMPUTACIONAIS	75
4.1 INSTÂNCIAS PARA TESTE	75
4.2 RESULTADOS DOS TESTES.....	79
4.2.1 Análise dos resultados pelos limitantes inferiores	79

4.2.2 Análise do tempo de processamento	89
4.2.3 Teste de robustez do <i>simulated annealing</i>	96
5 CONCLUSÕES E FUTUROS DESENVOLVIMENTOS	99
APÊNDICE A – Fórmulas auxiliares	106
ANEXO A – Análise dos parâmetros do <i>simulated annealing</i>	107
ÍNDICE	114

1 INTRODUÇÃO

Na versão clássica do problema de empacotamento unidimensional (*one dimensional bin packing problem* – 1-BPP) é dada uma lista $L = (a_1, a_2, \dots, a_n)$ de elementos (itens), uma lista de n de contentores com capacidade C e uma função $w(a_i) = w_i$ que dá o tamanho ou peso do item a_i satisfazendo a restrição $0 < w_i \leq C$, para $1 \leq i \leq n$. O problema consiste em alocar os itens ao menor número possível de contentores respeitando a restrição de que a soma dos pesos dos itens alocados aos contentores não ultrapasse a capacidade C do contentor. Ou seja, o problema consiste em particionar o grupo de itens no menor número de blocos, respeitando o peso total alocado a cada bloco (Coffman et al, 1997).

O primeiro trabalho realizado sobre o assunto foi elaborado por M. R. Garey e R. L. Graham nos Laboratórios Bell e por J. D. Ullman na Universidade de Princeton nos anos 1970. O segundo trabalho publicado alguns meses depois por D. S. Johnson estendeu os resultados estudando uma classe geral de algoritmos de soluções aproximadas para o problema. Em sua tese de Ph.D. no MIT, Johnson deu um tratamento mais detalhado ao problema em 1973 (Coffman et al, 1997), abrindo uma área de pesquisa muito rica e com as mais diversas aplicações dependendo do tamanho e forma dos itens, bem como da forma e capacidade dos contentores: problemas de corte e empacotamento (comuns na indústria); minimização de perdas de materiais grandes cortados em pequenos; alocação de memórias em computadores; carregamento de veículos com peso máximo a ser respeitado; agrupamento de propagandas em comerciais de televisão; etc. (Coffman et al, 1997).

Um conjunto de problemas semelhantes que pode ser considerado um caso particular dos BPP, são os problemas de Corte de Estoque (*Cutting Stock Problems* – CSP), onde a lista de itens possui itens com pesos repetidos. Desta forma os CSP são conhecidos como versões de alta multiplicidade dos BPP (Rietz et al, 2002).

Em alguns problemas de CSP as perdas podem ser estocadas para atendimento de demandas futuras e busca-se a otimização em períodos consecutivos (Trkman e

Gradisar, 2007). Em outros os objetivos podem ser vários, como no trabalho de Chu e Antonio (1999), que visa reduzir perdas no corte de materiais e também tempos de processamento dos cortes. Os autores apresentam melhores resultados que os obtidos manualmente em aplicações reais da indústria.

Nos BPP em geral os objetivos são: minimizar o número de contentores utilizados sem exceder as capacidades; minimizar as perdas e minimizar os tempos de processamento (Yesodha e Amudha, 2012).

Os BPP estão entre os mais importantes problemas de otimização combinatória, modelando uma grande variedade de aplicações reais. Por exemplo, o problema de carregamento de veículos em uma dimensão, respeitando o peso total da carga carregada pode ser enquadrado como um problema de corte e empacotamento (Jahromi et al, 2012).

Sendo um problema NP-Difícil, ainda não se conhece um algoritmo de solução para instâncias grandes que possa ser processado em tempo polinomial. Por isso muitos estudos encontrados na literatura apresentam heurísticas que geram boas soluções aproximadas, com tempos de processamento acessíveis à tomada de decisão, necessários em aplicações reais.

As primeiras soluções para o problema de corte e empacotamento - PCE foram criadas por Kantorovich em 1939 e Gilmore e Gomory em 1961, com a técnica de geração de colunas para resolução do problema de programação linear (Dantzig e Wolfe, 1960). A técnica de geração de colunas é uma variante do método simplex revisado que foi adotada originalmente para o problema contínuo relaxado *one-dimensional cutting stock problem* - 1D-CSP, isto é, um problema do 1D-CSP em que as restrições de integralidade das variáveis são relaxadas.

Na literatura são citados programas computacionais desenvolvidos para solucionar os PCE, como pode ser visto em Berberler et al (2010), mas tais programas têm como limitação as dimensões práticas encontradas nestes problemas, o que torna os tempos aplicados nos cálculos computacionais muito altos para aplicações reais

(Hoto et al, 2007). Além disso, tais programas computacionais geralmente têm como principal objetivo somente a redução de perdas (Cherri et al, 2014).

1.1 DEFINIÇÃO DO PROBLEMA E MOTIVAÇÃO

O objetivo de estudo nesta pesquisa consiste da resolução do problema de empacotamento unidimensional em um tempo computacional razoável para aplicações reais. Este problema visa a minimização da quantidade de contentores necessários para acomodar uma quantidade conhecida de itens.

Problemas de empacotamento são muito comuns na indústria, tanto na forma unidimensional, quanto bidimensional ou mesmo tridimensional. Algumas aplicações para esta classe de problemas, encontradas na literatura especializada, são o agendamento de comerciais em televisão, o carregamento de caixas em caminhões visando uma melhor utilização do volume ocupado, o corte de materiais visando uma melhor utilização de matéria-prima, e o planejamento de orçamento, entre outras (Hoto, et al 2007).

A relevância do tema tem atraído a atenção de profissionais de pesquisa operacional no desenvolvimento de modelos e métodos que solucionem instâncias reais, usualmente de grande porte, destes problemas. As indústrias, em geral, são estimuladas a solucionar seus problemas de corte e empacotamento, visando a redução de custos, de forma a se manterem competitivas no mercado.

1.2 OBJETIVOS PRINCIPAIS

O principal objetivo da pesquisa foi resolver o problema de empacotamento unidimensional de forma heurística e eficiente do ponto de vista computacional com tempos de processamento adequados para a tomada de decisão e com a geração de soluções de boa qualidade.

1.3 OBJETIVOS SECUNDÁRIOS

Alguns objetivos secundários também foram determinados, como a implementação de dez heurísticas simples para gerar a solução inicial em um mesmo ambiente computacional, e assim permitir a sua comparação em relação à qualidade da solução gerada e ao tempo necessário para o seu processamento.

Foram propostas novas heurísticas combinadas com heurísticas clássicas e a metaheurística *simulated annealing* que também foi modificada com a proposta de parâmetros específicos para cada instância do problema de empacotamento unidimensional.

1.4 MÉTODO ESCOLHIDO

O método proposto consiste de uma heurística de busca, que tem como solução inicial diversas heurísticas construtivas propostas na literatura, e descritas no capítulo 2. Também é proposta uma metaheurística *simulated annealing*, visando melhorar os resultados gerados pela heurística de busca, a qual é denominada de “Diferença de Quadrados” devido a forma das inequações que devem ser atendidas para gerar melhorias na função objetivo revisada do problema. Para avaliar a qualidade das soluções heurísticas, foram implementados os limitantes inferiores descritos na literatura. Os resultados ótimos para avaliação das heurísticas foram processados no programa computacional de otimização GUROBI 6.5.2.

1.5 ESTRUTURA DA DISSERTAÇÃO

Esta pesquisa está estruturada da seguinte forma: No capítulo 2 é feita uma revisão da literatura sobre limitantes inferiores e superiores; no capítulo 3 são apresentados o modelo matemático e as duas heurísticas propostas para solucionar o problema de empacotamento unidimensional melhorando as soluções iniciais geradas pelas heurísticas descritas no capítulo 2, uma delas baseada em relações de melhoria com a troca de itens entre contentores que provocam alterações na função objetivo revisada do problema e a outra na metaheurística *simulated annealing*, neste capítulo também são descritas as alterações propostas para os parâmetros do

simulated annealing; no capítulo 4 são apresentadas as instâncias utilizadas para teste e os testes computacionais realizados, e as conclusões e desenvolvimentos futuros são apresentadas no capítulo 5.

2 REVISÃO DA LITERATURA

Neste capítulo é feita uma revisão da literatura dos problemas de corte e empacotamento. Inicialmente, na seção 2.1 estão resumidas as tipologias de Wäscher et al (2007) e de Dyckhoff (1990). Na seção 2.2 estão resumidas as principais abordagens de métodos exatos e heurísticos utilizados para resolução dos problemas de corte e empacotamento. Na seção 2.3 são apresentados os limitantes inferiores propostos para o problema de empacotamento unidimensional que foram programados em C++ e que foram utilizados como critério de parada dos algoritmos de melhoria das soluções e para avaliação da qualidade das soluções por eles geradas, e na seção 2.4 estão detalhadas as heurísticas construtivas existentes, e que serão testadas nesta pesquisa.

2.1 TIPOLOGIA DOS PROBLEMAS DE CORTE E EMPACOTAMENTO

Uma tipologia é uma organização sistemática de objetos em categorias homogêneas com base em um dado conjunto de critérios. É orientada a objetos reais e “importantes”, enquanto que os hipotéticos e “menos importantes” são desconsiderados Wäscher et al (2007). Além de servir de base para as pesquisas orientadas, a tipologia também unifica definições e notações, facilitando a comunicação entre os pesquisadores, e ajuda a identificar áreas onde pouca pesquisa foi efetuada Wäscher et al (2007).

Os problemas de corte e empacotamento, que foram tratados nas tipologias de Wäscher et al (2007) e de Dyckhoff (1990), têm uma estrutura comum: São dados dois conjuntos de elementos, um conjunto de elementos de entrada (objetos) e um conjunto de elementos de saída (itens).

Alguns ou todos os itens devem ser agrupados em um ou mais conjuntos de modo a atender determinadas condições geométricas otimizando uma dada função objetivo Wäscher et al (2007).

Existem modelos baseados na associação de itens e em padrões de corte. Nos modelos baseados em padrões de corte, são determinadas as frequências nas quais

os itens devem ser alocados aos objetos (Cherri et al, 2014), porém nestes modelos não são considerados custos referentes à produção além da demanda e consequente formação de estoque nos itens, pois o plano de corte com menor perda não leva necessariamente ao menor custo total do processo por produzir itens com baixa demanda que acabam estocados por longos períodos.

No artigo de Dyckhoff (1990) é desenvolvida uma abordagem sistemática de compreensão da tipologia dos diversos tipos de problemas de corte e empacotamento fundamentada na estrutura lógica básica dos problemas e com o propósito de unificar o uso das diversas notações e direcionar os estudos futuros em tipos especiais de problemas e de preencher a lacuna existente da falta de artigos tratando da tipologia dos problemas. Os problemas de corte e empacotamento são caracterizados por terem as mesmas estruturas lógicas com nomes diferentes na literatura, como por exemplo:

- Corte (*cutting stock*) e minimização de perdas de material (*trim loss problems*).
- Empacotamento (*bin packing*), empacotamento dual (*dual bin packing*), empacotamento unidimensional (*strip packing*), empacotamento vetorial (*vector packing*) e problemas de mochila (*knapsack problems*).
- Carregamento de veículos (*vehicle loading*), carregamento de pallets (*pallet loading*), carregamento de contêineres (*container loading*) e carregamento de carros (*car loading*).
- *Assortment, depletion, design, dividing, layout, nesting e partitioning problems*.
- Gestão de capital (*capital budgeting*), cálculo de troco (*change making*), balanceamento de linhas (*line balancing*), alocação de memória (*memory allocation*) e problema de programação de múltiplos processadores (*multiprocessor scheduling problems*).

Estes problemas têm sido abordados na literatura sobretudo nas últimas décadas em diversas áreas do conhecimento como engenharia, ciências da computação e informação, matemática e pesquisa operacional (como uma abordagem interdisciplinar), segundo Dyckhoff (1990).

As estruturas lógicas comuns dos problemas de corte e empacotamento são as seguintes:

1. Dois grupos de dados cujos elementos são definidos por figuras geométricas em uma ou mais dimensões, sendo estes os objetos grandes disponíveis em estoque ou os itens pequenos, cuja demanda deve ser atendida.
2. Padrões de corte são combinações entre os itens pequenos que são alocadas aos objetos grandes, sendo as partes restantes diferentes dos itens pequenos, as perdas do corte.

Os problemas de empacotamento são caracterizados pelo preenchimento de espaços vazios em veículos, pallets, contêineres, contentores, etc. Estão fortemente relacionados devido à dualidade entre o material sólido e o espaço ocupado por ele (Dyckhoff, 1990).

Estes problemas podem estar relacionados com o preenchimento de dimensões não espaciais de diversas naturezas: problema da mochila (peso em uma dimensão); balanceamento de linha de montagem e programação de múltiplos processadores (dimensão temporal); orçamento de capital (para dimensões financeiras) e alocação de memória (para dimensões de dados) (Dyckhoff, 1990). Todos eles possuem características básicas dos problemas de empacotamento e podem ser considerados generalizações deles com restrições adicionais.

As características de tipos básicos dos problemas são classificadas de acordo com questões relativas a:

- O estoque dos objetos grandes.
- A lista dos itens pequenos.
- As combinações entre os itens pequenos para formar os objetos grandes.
- A alocação de itens pequenos aos padrões e estes aos objetos grandes.

A característica mais importante é a dimensionalidade, o menor número de dimensões dos números reais necessários para descrever a geometria dos padrões. Podendo ser de mais de três dimensões, geralmente quando é adicionado o tempo a

problemas de três dimensões. Um exemplo é o problema de empacotamento vetorial (*vector packing problem*) e a alocação de capital em múltiplos períodos (*multi-period capital budgeting*).

Outra característica importante é a medida de quantidade, que pode ser discreta ou contínua. Quando combinados problemas unidimensionais com medidas contínuas, estes são geralmente chamados de *one-and-a-half-dimensional*. As formas dos objetos e itens também é uma característica importante e relacionada a dimensionalidade. Ignorando as translações, uma figura é determinada por sua forma, tamanho e orientação. As figuras podem ser regulares (que podem ser descritas por poucos parâmetros) ou irregulares (que podem ser não convexas e simétricas e são comuns em diversas indústrias como a têxtil, de madeira e metais).

Um aspecto que torna os problemas mais fáceis ou difíceis de resolver é o tamanho relativo dos objetos e dos itens. Outra questão diz respeito ao agrupamento dos itens. A disponibilidade é outro aspecto importante e se refere a limitantes superiores e inferiores nas quantidades, sequência dos itens, ou data em que um objeto ou item pode ser cortado ou empacotado. A disponibilidade aparece quando os objetos e itens existem em certa ordem ou quando o tempo é uma característica do problema. Isso ocorre por exemplo quando contêineres têm datas de entrega ou barras precisam ser entregues sem atraso.

Nos problemas clássicos de corte e empacotamento existe um número infinito de objetos, todos com a mesma figura e tamanho. Carregamento de pallets (*pallet loading*) ou problemas de mochila (*knapsack problems*) são geralmente caracterizados por um único grande objeto.

Diferentes objetivos aparecem dependendo dos tipos a que se referem: quantidades dos objetos, geometria dos padrões, sequência ou combinação dos padrões. Os dados do problema podem ser determinísticos ou estocásticos, podem variar dentro de certas faixas de valores.

A classificação do problema tem importância na determinação da abordagem a ser utilizada para sua resolução.

As quatro características dos principais tipos são denotadas por símbolos que caracterizam os problemas:

1. Dimensionalidade: (1) Unidimensional; (2) Bidimensional; (3) Tridimensional; (N) N-dimensional com $N > 3$.
2. Tipo de associação: (B – de “Beladeproblem”) Todos os objetos e uma seleção de itens; (V – de “Verladeproblem”) Uma seleção de objetos e todos os itens.
3. Forma dos objetos grandes: (O) Um objeto; (I) Figuras idênticas; (D) Figuras diferentes.
4. Forma dos itens pequenos: (F) Poucos itens de diferentes figuras; (M) Muitos itens de muitas figuras diferentes; (R) Muitos itens de relativamente poucas figuras diferentes (não congruentes); (C) Figuras congruentes.

A tipologia de Wäscher et al (2007), introduziu os conceitos de maximização dos valores de saída e de minimização dos valores de entrada. Também foram alteradas as notações em alemão “*Beladeproblem*” – B e “*Verladeproblem*” – V, respectivamente.

Combinando os diferentes tipos de dimensionalidade, associação e arranjo é possível obter $4 \times 2 \times 3 \times 4 = 96$ tipos diferentes de problemas de corte e empacotamento denotados por quatro símbolos $\alpha/\beta/\lambda/\delta$. Assim por exemplo 3/B/O/F denota todos os problemas tridimensionais de corte e empacotamento onde um objeto deve ser alocado com a seleção de poucos itens pequenos. No tipo 1/V/I/F de carregamento de veículos, existe uma lista de itens pequenos e um número K também pequeno de objetos a ser minimizado.

Algumas vezes o valor dos objetos/itens pode ser assumido como diretamente proporcional ao seu tamanho (comprimento, área ou volume), nestes casos é possível converter problemas de maximização do valor de saídas e de minimização de valores de entradas em minimização de perdas, assim em alguns problemas utiliza-se o termo *trim-loss minimization*.

Outra modificação de Wäscher et al (2007) foi a inclusão dos formatos dos itens pequenos na tipologia, sendo eles de dois tipos: formatos regulares (como retângulos, círculos, caixas, etc.) e irregulares.

O tipo do problema deve ser considerado durante a escolha da abordagem de solução a ser utilizada, onde diferenciam as duas formas de abordagem: uma orientada a itens e objetos e outra orientada a padrões. No primeiro caso os itens são associados aos objetos diretamente. No segundo caso os itens são associados a padrões e estes associados aos objetos. A abordagem em padrões é geralmente mais eficiente quando o número de padrões é limitado ou há itens com grandes multiplicidades. Na Figura 1 estão os tipos básicos de problemas.

Figura 1 – Tipos básicos de problemas de corte e empacotamento
 Fonte: Wäscher et al (2007)

Os problemas de empacotamento podem ser classificados em três tipos, conforme Asgeirsson et al (2002):

- *On-line* – se os itens são alocados sequencialmente, sem levar em consideração os pesos dos itens não alocados. Ou seja, não há conhecimento dos pesos dos itens seguintes e não é possível mover um item de um contentor para outro.
- *Off-line* – se os pesos de todos os itens da instância são conhecidos, de modo que é possível mover um item de um contentor para outro.
- *Closed on-line* – é um tipo intermediário, no qual os pesos dos itens não são conhecidos a priori, mas a quantidade de itens é conhecida.

A dimensionalidade é uma das características mais importantes dos problemas e artigos tratando de problemas tridimensionais são mais raros e recentes na literatura.

A tipologia de Dyckhoff (1990) foi durante anos a principal organização sistemática dos problemas por seu conjunto de características, porém com o passar dos anos algumas deficiências foram identificadas nela. Baseados nestas deficiências e em algumas ideias originais de Dyckhoff, Wäscher et al (2007) propuseram uma tipologia melhorada com um novo critério de categorização, que define categorias de problemas de forma diferente da proposta por Dyckhoff (1990). Um novo sistema de nomes foi sugerido para as categorias de problemas, que diferente de Dyckhoff (1990) utiliza um sistema autoexplicativo em inglês, o idioma adotado internacionalmente para divulgação de pesquisas. O novo sistema de classificação foi testado com a literatura de corte e empacotamento entre os anos de 1995 e 2004, introduzindo entre outras melhorias a informação referente à forma dos itens (Wäscher et al, 2007).

2.2 ABORDAGENS EXATAS E HEURÍSTICAS PARA OS PROBLEMAS DE CORTE E EMPACOTAMENTO

Duas principais abordagens podem ser distinguidas: aquelas baseadas em heurísticas e as baseadas em relaxações de programação linear. As heurísticas são mais flexíveis, permitindo que mais características do problema sejam consideradas

(Dyckhoff, 1990). Quando programadas em sistemas de apoio a decisão, tais heurísticas permitem que a experiência do usuário seja levada em conta.

Vários artigos já foram publicados sobre o assunto e algumas heurísticas já foram propostas para tratá-lo. Em 1961, Gilmory e Gomory criaram a técnica de geração de colunas (*delayed column-generation technique* – DCGT). Abuhassan e Nasereddin (2011) citam o artigo de Gradisar et al (1999), que propuseram o procedimento de heurística sequencial (*sequential heuristic procedure* – SHP), que consiste na busca de um ponto de ótimo local em um conjunto de possíveis soluções para o problema, sendo este muitas vezes decomposto em subproblemas onde também a técnica DCGT é aplicada.

Eilon e Christofides (1971) resolveram o problema de forma exata, por meio do algoritmo *branch-and-bound* e programação dinâmica de problemas de carregamento de veículos. Também foram resolvidos de forma aproximada por uma heurística rápida denominada FFD (“*First Fit Decreasing*”) descrita detalhadamente na seção 2.4.

Alguns algoritmos utilizam outras abordagens, como a solução baseada na heurística SHP combinada com um procedimento de programação linear (Yang e Cui, 2010).

O número elevado de combinações, dependendo da heurística utilizada, pode tornar o problema inviável sob o aspecto computacional por demandar muito tempo para os cálculos. Portanto o tempo necessário para os cálculos do modelo também deve ser levado em consideração.

Segundo a definição de Wagner (1999), o problema pode ser considerado difícil quando o valor médio da razão entre a largura do objeto e a largura do item, C/w_i for menor que cinco; os outros problemas podem ser definidos como fáceis. Em geral problemas difíceis nesta definição têm altas perdas ou criam estoques indesejáveis nas últimas iterações de procedimentos como o SHP.

Em Wagner (1999), há referências a soluções baseadas em algoritmos genéticos, que são algoritmos robustos de buscas de soluções baseadas em seleção natural e mutações, em que uma população de soluções para o problema é melhorada por meio de mutações e trocas de partes dos bits gerados no modelo, sendo a estes atribuída uma probabilidade de sobrevivência associada ao valor da função objetivo, assim quanto melhor a solução, por mais iterações ela permanece no modelo.

O problema de empacotamento unidimensional (1-BPP) foi abordado em diversas formas incluindo procedimentos matemáticos, heurísticas determinísticas, metaheurísticas com inspiração biológica bem como no campo de outras metaheurísticas como a busca tabu e o *simulated annealing* (Sim et al, 2013).

Dell'Amico et al (2012) abordam uma variação do problema de empacotamento unidimensional, em que há precedência entre os itens (*bin packing problem with precedence constraints* – BPP-P) que modela muitos problemas de montagem, agendamento de tarefas e linhas de produção. No artigo o problema é abordado pela primeira vez com métodos exatos. Um critério de redução é desenvolvido, bem como um conjunto de limitantes inferiores, um método *branch-and-bound* e uma busca de vizinhança variável são implementados. Os testes computacionais são realizados em instâncias *benchmarking* da literatura e seus resultados são comparados aos das técnicas de programação linear inteira. São utilizadas as 269 instâncias SALBP disponíveis em <http://www.assembly-line-balancing.de>.

Extensões para os problemas de empacotamento são os chamados problemas de empacotamento generalizados – *generalized bin packing problems* – G-BPP, introduzido por Garey et al (1976) em que os contentores têm capacidades diversas. Outro problema relacionado aos BPP-P é o problema de balanceamento simples de linhas – *simple assembly line balancing problem* – SALBP. Neste problema as relações de precedência não são estritas.

Outros problemas relacionados aos problemas de empacotamento são os problemas de empacotamento com conflitos – *bin packing problems with conflicts* – BPPC, em que as relações de precedência são substituídas por relações de incompatibilidades

entre os itens (Dell'Amico et al, 2012). Os BPPC foram abordados por diversos autores como Gendreau et al (2004), Khanafer et al (2010) e Muritiba et al (2010).

Algumas aplicações para os BPPC consistem em problemas de agendamento de tarefas, alocação de processos a processadores, balanceamento de cargas de tarefas em paralelo e distribuição de alimentos em entregas *delivery* onde alguns itens não podem ser transportados no mesmo veículo e posterior roteirização das entregas (Muritiba et al, 2010). No artigo de Muritiba et al (2010) são apresentados novos limitantes inferiores e superiores e uma abordagem exata baseada em cobertura de conjuntos através de um algoritmo *branch-and-price*. Os testes computacionais foram realizados em 800 instâncias, obtendo soluções ótimas em 780 delas.

Capua et al (2015) abordam os BPPC com um método baseado na metaheurística *iterated local search* e um algoritmo de busca em vizinhança larga chamado *ejections chains* que mostrou ser competitivo em relação aos resultados de Muritiba et al (2010). Gendreau et al (2004) comparam seis heurísticas para resolver os BPPC e propõem um novo limitante inferior.

No problema de coloração de vértices (*vertex coloring problem*) – VCP – deseja-se minimizar o número de cores utilizadas para colorir vértices de tal modo que vértices vizinhos não tenham as mesmas cores. Uma aplicação consiste na coloração de mapas minimizando as cores utilizadas. Os problemas VCP são considerados versões generalizadas dos BPPC, e, portanto também dos BPP com pesos dos itens todos iguais a zero e com um grafo de ligação entre os itens (Muritiba et al, 2010).

Outra variação interessante dos problemas de empacotamento são os *ordered open end bin packing problems*, onde as ordens dos itens são dadas como entrada do problema. E o último item entre os alocados ao contentor pode exceder sua capacidade Dell'Amico et al (2012).

Sim et al (2013) apresentam uma hyper-heurística que tenta determinar qual heurística de um conjunto pré-determinado será a melhor escolha para uma dada

instância. No programa as heurísticas são selecionadas com base no percentual de itens com pesos dentro de intervalos de capacidades dos contentores.

O termo hyper-heurísticas originalmente descrito como “heurísticas para selecionar heurísticas” evoluiu para o campo de “heurísticas para gerar heurísticas”, usando programação genética para criar novas heurísticas através das partes constituintes. A origem do termo vem do trabalho de Fisher & Thompson usando técnicas para gerar combinações de heurísticas simples para construção de soluções para problemas de *job shop scheduling*.

Aktin e Özdemir (2009) sugerem um modelo em dois estágios, sendo que no primeiro são gerados os padrões de corte e no segundo os planos de corte. Por meio de dois modelos lineares os autores sugerem um modelo para resolver planos de corte na indústria.

Trabalhos como os de Burke et al (2010) utilizam hyper-heurísticas com algoritmos genéticos para resolver os BPP *on-line*. Os algoritmos têm acesso à memória de itens anteriores para determinar a melhor abordagem a ser utilizada nos itens futuros, sendo esta estratégia dinâmica um diferencial em relação às estratégias estáticas para este tipo de problema. Estratégia semelhante pode ser encontrada no trabalho de Sim e Hart (2013) utilizando uma forma compacta de programação genética para selecionar a melhor heurística para cada tipo de instância, caracterizando assim também uma hyper-heurística.

Metaheurísticas também são muito utilizadas para resolver tanto os CSP quanto os BPP. Otimização de colônia de formigas – ACO (*ant colony optimization*) foram propostas no trabalho de Levine e Ducatelle (2004) com e sem combinações com buscas locais. Os resultados reportados indicam que o algoritmo é comparável aos de Liang et al (2002) para os CSP e aos de Martello e Toth para os BPP. Porém são falhos quando comparados aos algoritmos genéticos HGGA de Falkenauer (1996), conforme Levine e Ducatelle (2004). Os autores indicam como trabalhos futuros comparar os algoritmos com outras instâncias e com outros algoritmos como *bin*

packing solution procedure – BISON de Scholl et al (1997) e escolhas dinâmicas para os parâmetros da ACO.

Liang et al (2002) utilizam programação evolucionária – EP com dois tipos de mutação para os CSP com e sem variação nas capacidades dos contentores. Os testes computacionais indicam que as EP provem alternativas mais simples e melhores que os algoritmos genéticos – GA.

Khuri et al (1995) apresentam uma heurística evolucionária com operadores genéticos específicos e testa o algoritmo em instâncias da biblioteca de pesquisa operacional (*OR-Library*) disponíveis por e-mail (Beasley, 1990).

Layeb e Chenche (2012) utilizam GRASP – *greedy randomized adaptive search procedure* em dois algoritmos para resolver os BPP: um baseado em uma hibridização de heurísticas como *Best Fit* - BF e *First Fit* - FF (descritas em detalhes na seção 2.4) e outro num algoritmo de busca tabu.

Cunha et al (2008) apresentam heurísticas para 1-BPP baseadas em GRASP e em busca de vizinhança variável – BVV motivados por sua aplicação em problemas de distribuição de produtos nas quais as entregas devem ser alocadas a veículos de modo a minimizar a frota utilizada respeitando-se as capacidades dos veículos. Tais problemas exigem programação diária de entregas e prazo reduzido para essa atividade, assim o tempo de processamento e a qualidade das soluções são aspectos importantes para a avaliação das heurísticas propostas (Cunha et al, 2008). Os autores utilizam a heurística MBS' de Fleszar e Hindi (2002) (detalhadas na seção 2.4) com diferentes sementes geradas aleatoriamente nos procedimentos. Foram utilizadas as instâncias de Falkenauer (1997) e Schwerin e Washer (1997).

As heurísticas obtiveram bons resultados nas instâncias Falkenauer U, Schwerin e Washer. Mas não nas Falkenauer T, o que segundo os autores não desqualifica estas instâncias porque elas não ocorrem em aplicações práticas de carregamento de veículos, visto que dificilmente somente três cargas serão somadas gerando exatamente a capacidade do veículo.

Muntean (2007) comparara resultados de programação genética (*genetic programming*) – GP com os da heurística BFD (*Best Fit Decreasing*) e verifica que em muitos casos os resultados dos GP superam os do BFD. A heurística BFD é descrita em detalhes na seção 2.4.

Gourgand et al (2014) abordam o problema HCT e o compara aos BPP. Os HCT podem ser vistos como extensões dos BPP com restrições de incompatibilidades. O método proposto no artigo consiste num algoritmo híbrido entre uma metaheurística e um algoritmo de lista (Next Fit - NF, First Fit - FF e Best Fit - BF) em dois espaços de soluções: um com uma lista de itens e outro que representa a alocação de cada item a um contentor. Os testes computacionais foram realizados nas instâncias de Falkenauer (1997) e os resultados são equivalentes aos de outros autores. Os algoritmos de lista NF, FF e BF são descritos na seção 2.4, partindo de instâncias com itens previamente classificados em ordem decrescente de peso, formando os algoritmos NFD, FFD e BFD.

Uma busca de vizinhança consistente é proposta por Buljubašić e Vasquez (2016) para resolver os BPP. Iniciando com uma solução com $m = UB - 1$ contentores, o algoritmo busca gerar uma solução viável com m contentores, onde UB é uma solução inicial gerada por uma variação do FF. O algoritmo remove dois contentores da solução e tenta realocar os itens destes contentores com o objetivo de minimizar o peso total dos itens não alocados com uma busca tabu. Soluções melhores ou equivalentes às publicadas na literatura são obtidas no conjunto Hard28 de instâncias de BPP, um dos conjuntos mais difíceis encontrados da literatura.

Suárez et al (2006) apresentam um algoritmo para 1-BPP cujo número de iterações necessárias para encontrar uma partição de itens que não mude é limitada pelo método point Jacob matrix. O algoritmo foi testado numa grande variedade de instâncias e foi bem-sucedido numa classe considerada difícil: os conjuntos de instâncias de Falkenauer (1996) e Scholl et al (1997).

Yesodha e Amudha (2012) fazem um comparativo entre *First Fit* - FF e *Best Fit* - BF em instâncias de diferentes níveis de dificuldade.

Martel (1985) apresenta um algoritmo rápido para BPP *off-line* com razão de desempenho de pior caso igual a $4/3$.

Békési et al (2000) descrevem um modelo de Martel (1985) com um algoritmo de tempo linear. O algoritmo é baseado na classificação dos itens por peso em categorias e alocação dos itens de acordo com estas categorias.

Cui et al (2008) apresentam um algoritmo de geração de padrões de itens agrupando os não alocados a contentores e suas frequências, e então os apagando da lista de itens não alocados. O algoritmo é processado até que a lista de itens não alocados torna - se vazia. Seus resultados mostram serem superiores a procedimentos de redução de padrões anteriores.

Um algoritmo de redução é proposto no artigo de Gabay et al (2014) para problemas de BPP e similares. As reduções são geralmente críticas para o sucesso de algoritmos de empacotamento.

Korf (2002) propõe um novo algoritmo para soluções ótimas para 1-BPP. Em vez de considerar os diferentes contentores em que cada item pode ser alocado, seu algoritmo considera as diferentes formas em que cada contentor pode ser preenchido. O autor informa que o algoritmo pode ser processado mil vezes mais rápido que outros com problemas de 60 itens. O algoritmo chamado de *bin completion* – BC explora um espaço de soluções diferente, o que o torna mais rápido que o algoritmo MTP de Martello e Toth (Korf, 2002).

Schreiber e Korf (2013) desenvolvem uma melhoria no algoritmo *Bin Completion* – BC de Korf (2003), criando o algoritmo *Improved Bin Completion* - IBC e relatam que o algoritmo proposto é cinco ordens de grandeza mais rápido no processamento que o algoritmo *branch-and-cut-and-price* - BCP em instâncias com relativamente poucos itens.

Alvim et al (2004) propõem uma heurística híbrida de melhoria para problemas de empacotamento com aplicação ao problema de agendamento de múltiplos processadores. A heurística consiste na utilização de novos limitantes inferiores e procedimentos de redução combinados com uma busca tabu e um procedimento de redistribuição de carga. Os resultados obtidos estão entre os melhores da literatura, com a obtenção de novas soluções ótimas em algumas instâncias. Os BPP foram resolvidos com um procedimento híbrido sofisticado de melhoria usando busca tabu (chamado HI_BP), encontrando soluções ótimas para 1370 instâncias *benchmark* da literatura. Os autores relatam que o algoritmo HI_BP melhorou a melhor solução conhecida de três instâncias (BPP56, BPP71 e BPP81) nos conjuntos Was1 e Was2.

Martello e Toth (1990) resolveram um BPP com um procedimento *branch-and-bound* (MTP). Schwerin e Wäsher (1997) desenvolveram um novo limitante inferior para os BPP baseados em problemas de corte de estoque, integrando o novo limitante inferior no MTP, obtendo resultados de muito boa qualidade no procedimento MTPCS. Os resultados de MTPCS são apresentados na página da WEB: <http://www.apdio.pt/sicup/Sicuphomepage/research.htm> e mostram que das 17 instâncias do conjunto Gau onde foi aplicado, o algoritmo encontrou a solução ótima para um problema, as melhores soluções conhecidas para oito problemas e falhou em encontrar a solução ótima em oito problemas (Loh et al, 2008).

Coffman et al (1997) fizeram uma revisão da literatura de métodos aproximados para o problema, incluindo análises de piores casos de algoritmos muito conhecidos como o FFD e o BFD detalhados na seção 2.4.

Bender et al (2001) propõem duas novas variantes para o algoritmo *Sum-of-Squares* - SS proposto por Csirik et al (1999) para problemas de empacotamento *on line*. A primeira variante é processada em tempo computacional $O(N\sqrt{B}\log B)$ e possui a mesma perda esperada do algoritmo original. A segunda variante possui tempo computacional $O(N\log B)$ e gera melhores resultados na maioria das distribuições de pesos dos itens tratadas no artigo (Bender et al, 2001). O algoritmo foi comparado

ao *Best Fit* - *BF* para problemas de alocação de memória. Os resultados indicaram que nenhum dos algoritmos é consistentemente melhor que o outro e que se a aleatoriedade nos pesos dos itens for baixa, *SS* parece ter menor desperdício que o *BF*, se a aleatoriedade é grande, *BF* possui menor desperdício que o *SS*. Csirik et al (2006) apresentam uma análise teórica e novas variantes da heurística *Sum-of-Squares* introduzida por Csirik et al (1999).

Wäscher e Gau (1996) tratam do problema de geração de soluções inteiras para os CSP. Os métodos heurísticos da literatura são comparados através da qualidade da solução e dos tempos de processamento computacionais. Foram utilizadas quatro mil instâncias para comparação e os autores identificaram duas heurísticas como melhores para geração de soluções inteiras.

Scholl et al (1997) desenvolveram um método híbrido *bin packing solution procedure* (BISON) que combina uma busca tabu com um procedimento de *branch-and-bound*. Carvalho (2002) desenvolve um algoritmo exato baseado em geração de colunas e *branch-and-bound*.

Fleszar e Hindi (2002) desenvolveram um algoritmo híbrido PMBS + VNS, onde PMBS denota a perturbação do MBS (*minimal bin slack*) de Gupta e Ho (1999) com um procedimento de busca em vizinhança variável (*variable neighborhood search*) - VNS. Um dos critérios de parada utilizados foi a obtenção do limitante inferior do contínuo e dos limitantes inferiores desenvolvidos por Fekete e Schepers (2001). O algoritmo foi testado com 1370 instâncias da literatura e obteve soluções ótimas em 1329 delas, sendo quatro delas melhores que as melhores soluções conhecidas na época. Os algoritmos MBS e os derivados dele são detalhados na seção 2.4.

No artigo de Loh et al (2008) a heurística *Weighted Annealing* – *WA*, uma nova metaheurística recentemente introduzida em artigos da literatura de física, foi utilizada para resolver os BPP. A ideia principal nessa heurística é mudar o panorama do problema fazendo uso da história de cada rodada da otimização (Loh et al, 2008). Os autores mudam o panorama do problema atribuindo pesos para diferentes partes do espaço de solução. Os pesos podem ser atribuídos

aleatoriamente ou dando ênfase para partes do espaço de solução que estejam mais longe da solução ótima.

WA compartilha características com outras metaheurísticas como *simulated annealing* – SA e *deterministic annealing* – DA. Em particular SA e DA começam com uma solução inicial e fazem uma sequência de movimentos que podem eventualmente piorar a solução. WA, diferente de SA e DA, considera valores não somente da função objetivo, mas também informações sobre como partes do espaço de solução estão sendo resolvidas a cada iteração do procedimento (Loh et al, 2008). A característica chave do procedimento é a distorção dos pesos dos itens que permitem movimentos para cima e para baixo. As mudanças nos pesos aparentes dos itens são obtidas através de assimilações de diferentes pesos para os contentores e os itens de acordo com o quão bem os contentores estão montados, utilizando a mesma função objetivo considerada por Fleszar e Hindi (2002) que procura reduzir o número de contentores por meio da maximização da soma dos quadrados das cargas dos contentores.

O algoritmo proposto por Loh et al, (2008) é denotado por WABP – *weighted annealing bin packing*, que começa por uma solução inicial gerada por um algoritmo de FFD – *First Fit Decreasing* e para cada contentor é calculada a carga e a capacidade residual. No início do processamento o parâmetro de temperatura é alto, permitindo mais facilmente os movimentos dos itens. O parâmetro de temperatura é reduzido a cada iteração e o problema fica mais parecido com o original. A cada iteração se um movimento de inserção de um item em outro contentor gera um contentor vazio e o limitante inferior é atingido, o algoritmo para, caso o contrário é processado o movimento de troca de itens entre contentores. Se o movimento não produz melhorias, o movimento de troca de um item de um contentor com um par de itens de outro contentor é processado e caso este não melhore a solução, o movimento de troca entre contentores de dois pares de itens é processado.

Os resultados dos experimentos computacionais foram comparados aos da literatura de outros algoritmos (HI_BP, PMBS' + VNS, BISON e MTPCS) e mostram que os algoritmos HI_BP e WABP encontraram soluções ótimas para 1370 instâncias. O

algoritmo PMBS' + VNS encontrou soluções ótimas para 1329 instâncias (Loh et al, 2008). A comparação dos resultados mostrou que das 1587 instâncias, WABP encontrou a solução ótima ou a melhor solução conhecida em 1584 delas e gerou novas soluções ótimas para três instâncias. O tempo médio de processamento das instâncias foi de 0,16 segundos, e que WABP possui melhor desempenho que algoritmos sofisticados como busca tabu (HI_BP), VNS (PMBS'+VNS) e *branch and bound* com *cutting stock lower bound* - MTPCS (Loh et al, 2008).

2.3 LIMITANTES INFERIORES

Esta seção do trabalho detalha os principais métodos utilizados para a obtenção de limitantes inferiores para o número de contentores de uma solução para os BPP. Todos os limitantes descritos nesta seção foram implementados nesta pesquisa para critério de parada dos algoritmos e para avaliação dos limitantes superiores.

A solução ótima de qualquer instância I de BPP está limitada entre o limitante inferior L_1 (descrito nas próximas seções) e o número de itens da instância N (Gent, 1998):

$$L_1 \leq Opt(I) \leq N \quad (1)$$

De modo que quanto mais próximos forem os valores dos limitantes inferiores dos limitantes superiores, mais próxima a solução estará do ótimo. Em particular quando o limitante inferior for igual ao limitante superior (qualquer solução válida), o limitante superior é uma solução ótima para o problema.

2.3.1 O limitante inferior de Johnson

A heurística *First Fit Decreasing* - FFD ordena os itens em ordem decrescente de peso e aloca cada item ao primeiro contentor onde houver capacidade disponível. Já a heurística *Best Fit Decreasing* - BFD ordena os itens em ordem decrescente de peso e aloca cada item ao contentor que ficar com a maior capacidade ocupada após a alocação. As heurísticas FFD e BFD são descritas em detalhes na seção 2.4.

Pela relação de pior caso de Johnson (1974), temos que a solução gerada pela heurística FFD satisfaz a relação:

$$FFD(I) \leq \frac{11}{9} z(I) + 4 \quad (2)$$

Onde $z(I)$ e $FFD(I)$ são respectivamente a solução ótima e a solução gerada pela heurística FFD para a instância I . Esta relação também é válida para a heurística BFD. Desta forma um limitante inferior simples pode ser obtido com algoritmos aproximados como o BFD e FFD, sendo este conhecido por limitante inferior da *Best Fit* L_{BF} (Marterllo e Toth, 1990):

$$L_{BF}(I) = \frac{9}{11} (BFD(I) - 4) \quad (3)$$

2.3.2 O limitante inferior L_1 do contínuo

Um limitante inferior imediato para o problema de empacotamento unidimensional é dado pelo limitante inferior do contínuo L_1 , sendo este o arredondamento para o maior número inteiro maior ou igual à soma dos pesos dos itens da instância I , dividido pela capacidade C dos contentores.

É o limitante inferior mais utilizado na literatura devido a sua simplicidade, dado por:

$$L_1 = \left\lceil \sum_{i \in I} \frac{w_i}{C} \right\rceil \quad (4)$$

2.3.3 O limitante inferior L_2 de Martello e Toth (1990)

Em seu estudo, Martello e Toth (1990) apresentam novos limitantes inferiores para o problema. Um deles, L_2 , descrito nesta seção é baseado no agrupamento de itens em conjuntos por faixas de peso limitadas por divisões da capacidade dos contentores e outro L_3 descrito na seção seguinte que é baseado num procedimento de redução das instâncias que também foi proposto pelos autores Martello e Toth (1990).

O limitante inferior L_2 é utilizado por diversos autores na literatura de problemas de empacotamento unidimensional.

Dado um número inteiro θ , com $0 \leq \theta \leq \frac{1}{2}C$, o conjunto I dos itens da instância pode ser dividido em três subconjuntos de acordo com o peso w_i do item, sendo:

$$N_1 = \{i \in I: w_i > C - \theta\} \quad (5);$$

$$N_2 = \{i \in I: C - \theta \geq w_i > \frac{1}{2}C\} \quad (6);$$

$$N_3 = \{i \in I: \frac{1}{2}C \geq w_i \geq \theta\} \quad (7);$$

Como os itens em $N_1 \cup N_2$ têm pesos maiores que metade da capacidade dos contentores, cada um dos itens requer um contentor separado. Então $|N_1| + |N_2|$ contentores são necessários para acomodar estes itens. Nenhum dos itens em N_3 pode ser alocado a um contentor contendo os itens do conjunto N_1 devido a restrição de capacidade. A capacidade ociosa dos $|N_2|$ contentores necessários para alocar os itens do conjunto N_2 é $\bar{C} = |N_2|C - \sum_{i \in N_2} w_i$.

No melhor caso os itens de N_3 irão preencher completamente as capacidades ociosas dos $|N_2|$ contentores, e os pesos dos itens remanescentes $S = \sum_{i \in N_3} w_i - \bar{C}$, irão requerer $\left\lceil \frac{S}{C} \right\rceil$ novos contentores.

Então, para cada inteiro θ , $L(\theta)$ é um limitante inferior do número m de contentores necessários:

$$L(\theta) = |N_1| + |N_2| + \max\left(0, \left\lceil \frac{\sum_{i \in N_3} w_i - (|N_2|C - \sum_{i \in N_2} w_i)}{C} \right\rceil\right) \quad (8)$$

Logo:

$$L_2 = \max(L(\theta)) \quad (9)$$

É também um limitante inferior válido para m .

Considerando $\theta = 0$, temos $|N_1| = 0$ e:

$$\begin{aligned} L(0) &= 0 + |N_2| + \max\left(0, \left\lceil \frac{\sum_{i \in N_3} w_i - (|N_2|C - \sum_{i \in N_2} w_i)}{C} \right\rceil\right) \\ &= |N_2| + \max\left(0, \left\lceil \frac{\sum_{i \in I} w_i - |N_2|C}{C} \right\rceil\right) = |N_2| + \max(0, \lceil L_1 - |N_2| \rceil) \end{aligned} \quad (10)$$

Segue que $L_2 \geq L(0) = \max(|N_2|, L_1)$, logo:

$$L_2 \geq L_1 \quad (11)$$

O pseudocódigo do procedimento de cálculo do limitante L_2 é apresentado na Figura 2.

```

1: Procedimento  $L_2(n, w_j, L, C)$ 
2:  $N = \{1, \dots, n\}$ 
3:  $j^* = \min\{j \in N : w_j \leq C/2\}$ 
4: Se  $j^* = 1$  Então  $L_2 = \lceil \sum_{j=1}^n w_j / C \rceil$ 
5: Senão
6:    $CJ12 = j^* - 1$  (comentário:  $CJ12 = |J_1| + |J_2|$ )
7:    $SJ^* = \sum_{j=j^*}^n w_j$ 
8:    $j' = \min\{j \in N : j < j^* \text{ e } w_j \leq C - w_{j^*}\}$  ( $j' = j^*$  se não há  $w_j$  nestas condições)
9:    $CJ2 = j^* - j'$  (comentário:  $CJ2 = |J_2|$ )
10:   $SJ2 = \sum_{j=j'}^{j^*-1} w_j$  (comentário:  $SJ2 = \sum_{j \in J_2} w_j$ )
11:   $j'' = j^*$ 
12:   $SJ3 = w_{j''}$ 
13:   $w_{n+1} = 0$ 
14:  Enquanto  $w_{j''+1} = w_{j''}$  Faça
15:     $j'' = j'' + 1$ 
16:     $SJ3 = SJ3 + w_{j''}$ 
17:    (comentário:  $SJ3 = \sum_{j \in J_3} w_j$ )
18:  Fim enquanto
19:   $L_2 = CJ12$ 
20:  Enquanto  $j'' > N$  ou  $CJ12 + [(SJ^* + CJ2)/C - CJ2] \leq L_2$ 
21:     $L_2 = \max(L_2; CJ12 + [(SJ3 + SJ2)/C - CJ2])$ 
22:     $j'' = j'' + 1$ 
23:    Se  $j'' \leq N$  Então
24:       $SJ3 = SJ3 + w_{j''}$ 
25:      Enquanto  $w_{j''+1} = w_{j''}$  Faça
26:         $j'' = j'' + 1$ 
27:         $SJ3 = SJ3 + w_{j''}$ 
28:      Fim enquanto
29:      Enquanto  $j' > 1$  e  $w_{j'-1} \leq C - w_{j''}$  Faça
30:         $j' = j' - 1$ 
31:         $CJ2 = CJ2 + 1$ 
32:         $SJ2 = SJ2 + w_{j'}$ 
33:      Fim enquanto
34:    Fim se
35:  Fim enquanto
36: Fim se
37: Saída ( $L_2$ )

```

Figura 2 – Pseudocódigo do cálculo do limitante inferior L_2 de Martello e Toth
Fonte: Martello e Toth (1990)

Se os itens em I estiverem ordenados em ordem decrescente de peso, L_2 pode ser calculado em tempo $O(N)$, gerando um conjunto D com os valores distintos de $w_i \leq C/2$ e em seguida, se $D = \emptyset$, então $L_2 = |N_1| + |N_2|$, caso o contrário $L_2 = \max_{\theta \in D} (L(\theta))$.

2.3.4 O procedimento de redução de Martello e Toth

Um subconjunto de itens F contido no conjunto N é dito viável se $\sum_{j \in F} w_j \leq C$. Dados dois subconjuntos F_1 e F_2 , dizemos que F_1 domina F_2 , se o valor da solução ótima obtido impondo para um contentor, dito i^* , os valores $x_{i^*j} = 1$ (se $j \in F_1$) e $x_{i^*j} = 0$ (caso o contrário), não é maior que o valor que pode ser obtido forçando os valores $x_{i^*j} = 1$ (se $j \in F_2$) e $x_{i^*j} = 0$ (caso o contrário). Se os itens de um conjunto F dominam todos os outros, o conjunto F pode ser adicionado a um novo contentor e removido do conjunto N .

```

1: procedimento MTRP ( $n, w_j, L, C$ )
2:  $N = \{1, \dots, n\}$ 
3:  $\bar{N} = \emptyset$ 
4:  $z^r = 0$ 
5: Para  $j = 1$  até  $n$  faça  $b_j = 0$ 
6: Enquanto  $N \setminus \bar{N} \neq \emptyset$ 
7:   encontrar  $j = \min\{h: h \in N \setminus \bar{N}\}$ 
8:    $N' = N \setminus \{j\} = \{j_1, \dots, j_l\}$  com  $w_{j_1} \geq \dots \geq w_{j_l}$ 
9:    $F = \emptyset$ 
10:  encontrar o maior valor de  $k$  tal que  $w_j + \sum_{q=l-k+1}^l w_{j_q} \leq C$ 
11:  Se  $k = 0$  então  $F = \{j\}$ 
12:  Senão
13:     $j^* = \min\{h \in N': w_j + w_h \leq C\}$ 
14:    Se  $k = 1$  ou  $w_j + w_{j^*} = C$  então  $F = \{j, j^*\}$ 
15:    Senão  $k = 2$  então
16:      encontrar  $j_a, j_b \in N'$ , com  $a < b$ , tal que
17:       $w_{j_a} + w_{j_b} = \max\{w_{j_r} + w_{j_s}: j_r, j_s \in N', w_j + w_{j_r} + w_{j_s} \leq C\}$ 
18:      Se  $w_{j^*} > w_{j_a} + w_{j_b}$  então  $F = \{j, j^*\}$ 
19:      Senão  $w_{j^*} = w_{j_a}$  e  $((b - a < 2)$  ou  $(w_j + w_{j_{b-1}} + w_{j_{b-2}} > C))$ 
20:      Então  $F = \{j, j_a, j_b\}$ 
21:      Fim se
22:    Fim se
23:    Se  $F = \emptyset$  então  $\bar{N} = \bar{N} \cup \{j\}$ 
24:    Senão
25:       $z^r = z^r + 1$ 
26:      Para cada  $h \in F$  faça  $b_h = z^r$ 
27:       $N = N \setminus F$ 
28:    Fim se
29:  Fim Enquanto
30: Saída ( $z^r, b_j$ )

```

Figura 3 – Pseudocódigo do procedimento de redução MTRP de Martello e Toth
Fonte: Martello e Toth (1990)

Checar todas as combinações é inviável em situações práticas devido ao elevado número de combinações a serem avaliadas, por isso os autores propuseram um algoritmo chamado *Martello & Toth Reduction Procedure* – MTRP, que procura por conjuntos F de até 3 itens e para cada item j procura pela existência de um conjunto F tal que para cada $j \in F$ domina todos os outros conjuntos que contém o item j .

Sempre que conjuntos desta forma são encontrados, eles são alocados a um novo contentor, eliminados do espaço de busca e o processo continua com os demais itens. Reduzindo desta forma o problema e transformando-o em um novo problema de empacotamento menor. A complexidade de MTRP é $O(N^2)$. Este procedimento pode ser utilizado para o cálculo de um novo limitante inferior L_3 .

2.3.5 O limitante inferior L_3 de Martello e Toth

Na Figura 4 está detalhado o pseudocódigo para o cálculo do limitante inferior L_3 .

```

1: procedimento  $L_3(n, w_j, L, C)$ 
2:  $L_3 = 0$ 
3:  $z = 0$ 
4:  $\bar{n} = n$ 
5: Para  $j = 1$  até  $n$  faça  $\bar{w}_j = w_j$ 
6: Fim Para
7: Enquanto  $\bar{n} \geq 1$  faça
8:   MTRP( $\bar{n}, \bar{w}_j, C$ )
9:    $z = z + z^r$ 
10:   $k = 0$ 
11:  Para  $j = 1$  até  $\bar{n}$  faça
12:    Se ( $\bar{b}_j = 0$ ) Então
13:       $k = k + 1$ 
14:       $\bar{w}_k = \bar{w}_j$ 
15:    Fim se
16:     $\bar{n} = k$ 
17:    Se ( $\bar{n} = 0$ ) Então
18:       $L_2 = 0$ 
19:    Senão ( $L_2(\bar{n}, \bar{w}_j, C)$ )
20:    Fim se
21:     $L_3 = \max(L_3, z + L_2)$ 
22:     $\bar{n} = \bar{n} - 1$  comentário: remoção do menor item de  $\bar{n}$ 
23:  Fim Para
24: Fim Enquanto
25: Saída ( $L_3$ )

```

Figura 4 – Pseudocódigo do cálculo do limitante inferior L_3 de Martello e Toth

Fonte: Martello e Toth (1990)

Após a execução do procedimento de redução MTRP, detalhado no pseudocódigo da Figura 3, para uma instância I de um problema de empacotamento, fazendo z_1^r denotar o valor de z^r e $I(z_1^r)$ a instância residual definida pelos itens $\{j \in N: b_j = 0\}$, o limitante inferior para I é dado por $z_1^r + L(I(z_1^r))$, onde $L(I(z_1^r))$ é o valor de qualquer limitante inferior para $I(z_1^r)$.

Supondo que $I(z_1^r)$ é relaxada de alguma maneira, por exemplo, com a eliminação de algum dos itens, e o procedimento MTRP é aplicado novamente para a instância residual relaxada, produzindo um número de contentores z_2^r e a instância relaxada $I(z_1^r, z_2^r)$, um limitante inferior para I é dado por $z_1^r + z_2^r + L(I(z_1^r, z_2^r))$. Iterando o processo, os autores obtiveram uma série de limitantes inferiores da forma (Martello e Toth, 1990):

$$L_3 = z_1^r + z_2^r + L(I(z_1^r, z_2^r)) \quad (12)$$

2.3.6 Melhor limitante inferior utilizado

O procedimento para cálculo dos limitantes inferiores consiste no cálculo do maior limitante inferior entre os quatro implementados:

$$L_{BPP} = \max(L_{BF}, L_1, L_2, L_3) \quad (13)$$

Na Figura 5 está descrito o pseudocódigo de cálculo dos limitantes inferiores, que tem como uma de suas entradas o limitante superior LS_{BF} que é o número de contentores da solução gerada pela heurística *Best Fit Decreasing* – BFD descrita na seção 2.4.

```

1: procedimento  $L_{BPP}(N, w_j, L, C, LS_{BF})$ 
2:  $L_{BF}(N, w_j, L, C)$ 
3:  $L_1(N, w_j, L, C)$ 
4:  $L_2(N, w_j, L, C)$ 
5: Se  $LS_{BF} = L_2$  Então  $L_{BPP} = L_2$ 
6: Senão  $L_3(N, w_j, L, C)$ 
7:    $L_{BPP} = \max(L_{BF}, L_1, L_2, L_3)$ 
8: Fim Se
9: Saída ( $L_{BPP}$ )

```

Figura 5 – Pseudocódigo do cálculo do limitante inferior L_{BPP}

Em todos os testes realizados a seguinte relação foi verificada:

$$L_{BF} \leq L_1 \leq L_2 \leq L_3 \quad (14)$$

Deste modo se o limitante superior for igual ao limitante L_2 , o limitante L_3 não precisa ser calculado, reduzindo o tempo de processamento, pois este limitante é o que depende o maior tempo de processamento.

2.4 LIMITANTES SUPERIORES

Esta seção do trabalho detalha as principais heurísticas para obtenção de limitantes superiores para o número de contentores do problema, descrevendo as principais heurísticas utilizadas na literatura para obtenção de soluções viáveis (SSD, MBS, MBS', MBSR, FFD, LFD, BFD, WFD e NFD) e WS (proposta nesta pesquisa como uma solução muito ruim para verificar o efeito das heurísticas para melhoria de solução apresentadas nesta pesquisa).

2.4.1 Heurística *Sum-of-Squares* - SSD

A heurística *Sum-of-Squares* foi proposta no trabalho de Csirik *et al* (1999).

Seu objetivo é resolver problemas de empacotamento *on-line*, e foi modificada nesta pesquisa para gerar soluções para problemas de empacotamento *off-line*.

A heurística busca alocar os itens aos contentores seguindo a regra: Alocar o item ao contentor que minimize a soma quadrática da quantidade de itens alocados ao contentor respeitando suas capacidades.

Essa regra pode ser traduzida da seguinte maneira: Sendo C a capacidade dos contentores e $N_g(g)$ o número de contentores em que a solução corrente tenha capacidade $C - gap_g$, $1 \leq gap_g < C$, ou seja o contentor tem ociosidade de tamanho gap_g . O próximo item deve ser alocado de tal forma que minimize $\sum_{1 \leq i < C} (N_g(gap_g))^2$. A regra pode ser aplicada a distribuições inteiras de pesos dos itens ou a distribuições que possam ser escalonadas a números inteiros dos pesos dos itens.

Na Figura 6 é apresentado o pseudocódigo do algoritmo SSD implementado nesta pesquisa.

```

1: procedimento SSD ( $N, w_i, L$ )
2:  $k = 0$ 
3:  $ss_p = 0$ 
4:  $gap_g = 0$ 
5:  $bin\Delta N_{g_{max}}(gap_g) = 0$ 
6:  $\Delta N_{g_{max}}(gap_g) = 0$ 
7: Para cada item  $i \in L$ 
8:   Para cada contentor  $j \in K$ 
9:      $l_j = \text{nível\_do\_bin}(j)$ 
10:    Se ( $l_j + w_i = C$ ) Então (comentário: ocupa completamente o
contentor)
11:       $ss_p = ss_p - 2N_g(w_i)$ 
12:       $k = j$ 
13:      Pare
14:    Fim se
15:     $gap_g = C - l_j$  (comentário: procura o contentor que maximiza o
valor de  $N_g(gap_g) - N_g(gap_g - w_i)$ )
16:    Se ( $l_j + w_i < C$  e  $N_g(gap_g) - N_g(gap_g - w_i) > \Delta N_{g_{max}}(gap_g)$ ) Então
17:       $\Delta N_{g_{max}}(gap_g) = N_g(gap_g) - N_g(gap_g - w_i)$ 
18:       $bin\Delta N_{g_{max}}(gap_g) = j$ 
19:    Fim se
20:  Fim Para
21:  Se ( $k = 0$  and  $\Delta N_{g_{max}}(gap_g) > 0$ ) então
22:     $ss_p = ss_p + 2(1 - \Delta N_{g_{max}}(gap_g))$ 
23:     $k = bin\Delta N_{g_{max}}(gap_g)$ 
24:  Fim se
25:  Se ( $k = 0$  e  $\Delta N_{g_{max}}(gap_g) = 0$ ) então
26:     $K = K + 1$ 
27:     $k = K$ 
28:     $ss_p = ss_p + 1 + 2N_g(C - w_i)$ 
29:  Fim se
30:   $l_k += w_i$ 
31:   $S_H(k) = S_H(k) \cup \{i\}$ 
32:   $k = 0$ 
33:   $bin\Delta N_{g_{max}}(gap_g) = 0$ 
34: Fim Para
35: Saída ( $K, S_H$ )

```

Figura 6 – Pseudocódigo da heurística Sum-of-Squares – SSD

2.4.2 Heurística *Minimal Bin Slack* – MBS

A heurística *Minimal Bin Slack*, proposta por Gupta e Ho (1999), é focada na ocupação dos contentores. A cada iteração da heurística uma tentativa é feita para preencher a capacidade do contentor o máximo possível. A cada estágio do algoritmo, uma lista Z' com os n' itens ainda não alocados a contentores é mantida em ordem decrescente dos pesos dos itens.

```

1: procedimento MBS( $q$ )
2: Para  $r = q$  até  $n'$  faça
3:    $i = Z'_r$ 
4:   Se ( $w_i \leq s(A)$ ) então
5:      $A = A \cup \{i\}$ 
6:     MBS( $q$ )
7:      $A = A \setminus \{i\}$ 
8:     Se ( $s(A^*) = 0$ ) então
9:       Fim
10:    Fim se
11:  Fim se
12:Fim Para
13:Se ( $s(A) < s(A^*)$ ) então
14:   $A^* = A$ 
15:Fim se
16:Saída ( $K, S_H$ )

```

Figura 7 – Pseudocódigo da heurística *Minimal Bin Slack* – MBS de Gupta e Ho
 Fonte: Fleszar e Hindi (2002)

Cada vez que um conjunto de itens é formado e alocado a um contentor, estes são retirados da lista Z' , mantendo a ordem decrescente dos pesos dos itens e o processo de alocação recomeça com um novo item. O algoritmo termina quando a lista Z' se torna vazia (Fleszar e Hindi, 2002).

O procedimento é programado de forma recursiva e é apresentado na Figura 7. A busca é iniciada com $q = 1$ e $A = A^* = \emptyset$, onde q é o índice do item na lista Z' , A é o conjunto de itens associados ao agrupamento atual e A^* é o conjunto de itens no melhor agrupamento. O espaço vazio no contentor é denotado por $s(A)$ e o número de itens na lista Z' é n' . $s(A)$ é iniciado com o valor C da capacidade dos contentores e atualizado cada vez que um item é adicionado ou removido do conjunto A .

Os resultados de Gupta e Ho (1999) indicam que MBS é superior em qualidade que outras heurísticas como FFD e BFD, descritas nas próximas seções, e que encontra soluções ótimas para instâncias em que uma solução com dois contentores existe (Fleszar e Hindi, 2002).

O algoritmo pode ser processado com tempos razoáveis, embora existam instâncias que levem horas de processamento antes de encontrar uma solução. Um exemplo é uma instância com capacidade dos contentores ímpar e pesos dos itens todos pares. Isso força o algoritmo a testar todas as combinações, visto que nenhum agrupamento de itens com folga zero existe (Fleszar e Hindi, 2002). Nas seções seguintes são detalhadas as modificações no algoritmo MBS de Gupta e Ho (1999) que foram propostas por Fleszar e Hindi (2002) para melhorar os tempos de processamento da heurística original.

2.4.3 Heurística *Minimal Bin Slack* modificada – MBS'

A heurística *Minimal Bin Slack modificada* – MBS' (Fleszar e Hindi, 2002), é uma versão modificada da heurística MBS de Gupta e Ho (1999). O algoritmo é processado da mesma forma que o original MBS descrito na Figura 7, mas com uma modificação. A modificação consiste na escolha de um item (semente) que deve ser obrigatoriamente alocado independente dos demais itens. Uma boa escolha para este item é o item de maior peso, o primeiro da lista. Isso deixa o menor espaço a ser preenchido no contentor, o que reduz o tempo de processamento (Fleszar e Hindi, 2002). Desta forma o algoritmo é sempre processado com $q = 2$ no pseudocódigo da Figura 7. Os autores citam um exemplo interessante. Considerando uma instância com três itens de pesos $w_1 = w_2 = w_3 = 5$ e três itens de pesos $w_4 = w_5 = w_6 = 3$ e capacidade dos contentores $C = 9$. Quando o algoritmo MBS é aplicado, ele gera uma solução com um contentor completamente ocupado com os três itens de peso 3. Como os pesos dos demais itens são maiores que metade das capacidades dos contentores, cada um deles ocupa um único contentor, o que gera uma solução final com 4 contentores, deixando os itens mais pesados e difíceis de alocar para as últimas iterações do algoritmo, o que aumenta o tempo de processamento.

No entanto quando a heurística MBS' é aplicada, ela fixa o item de maior peso não alocado ao contentor o que gera uma solução com três contentores com um item de peso 5 e um de peso 3 cada. Mantendo itens de menor peso, e, portanto, mais flexíveis para o final do algoritmo para preencher os espaços vazios dos contentores que já contém os itens de maiores pesos.

2.4.4 Heurística *Minimal Bin Slack Relaxed* – MBSR

A heurística *Minimal Bin Slack Relaxed* - MBSR (Fleszar e Hindi, 2002) é também uma versão modificada da heurística MBS de Gupta e Ho (1999). A modificação consiste na aceitação de agrupamentos de itens com folga positiva, ou seja, o algoritmo não tenta todas as combinações para formar um agrupamento de itens que preenche completamente o contentor, aceitando um agrupamento com uma folga pré-determinada, sendo isso feito por uma alteração no critério de parada do algoritmo MBS. A folga utilizada nesta pesquisa foi de 5% da capacidade dos contentores.

2.4.5 Heurística *First Fit Decreasing* - FFD

Uma das heurísticas simples mais populares e utilizadas por diversos autores como Sonuc et al (2017). O procedimento consiste em ordenar os itens em ordem decrescente de peso e em seguida alocar o item corrente ao primeiro contentor onde houver capacidade disponível.

Todos os contentores permanecem abertos para a alocação de itens durante todo o tempo de execução até que toda a sua capacidade esteja ocupada. Se não houver nenhum contentor com capacidade disponível para a alocação do item, um novo contentor é aberto. Dada uma instância com itens de pesos 82, 43, 40, 15, 12, 6 e contentores com capacidade igual a 100. A heurística FFD irá alocar o item com peso 82 ao primeiro contentor, o de peso 43 no segundo contentor, o de peso 40 no segundo contentor, o de peso 15 no primeiro contentor, o de peso 12 no segundo contentor e o último item de peso 6 no terceiro contentor. Gerando uma solução com

um total de 3 contentores, que é um contentor a mais que a solução ótima para esta instância (Korf, 2002).

Na Figura 8 é descrito o pseudocódigo da heurística FFD.

```

1: procedimento FFD ( $N, w_i, L$ )
2:  $K = 0$ 
3: Para cada item  $i \in L$ 
4:     Para cada contentor  $j \in K$ 
5:          $l_j = \text{nível\_do\_bin}(j)$ 
6:         Se ( $l_j + w_i \leq C$ ) Então
7:              $l_j += w_i$ 
8:              $S_H(j) = S_H(j) \cup \{i\}$ 
9:         Pare
10:    Senão
11:        Se ( $j = K$ ) então
12:             $K += 1$ 
13:             $S_H(K) = \{i\}$ 
14:        Fim se
15:    Fim se
16: Fim Para
17: Fim Para
18: Saída ( $K, S_H$ )
  
```

Figura 8 – Pseudocódigo da heurística *First Fit Decreasing* – FFD

2.4.6 Heurística *Last Fit Decreasing* - LFD

Semelhante ao FFD, a heurística *Last Fit Decreasing* – LFD consiste em ordenar a lista de itens em ordem decrescente de peso e alocá-los ao primeiro contentor onde houver capacidade disponível, porém iniciando a busca pelo último contentor aberto. Abre-se um novo contentor caso nenhum atenda a restrição de capacidade.

A heurística LFD funciona da mesma forma que a FFD, porém lendo os contentores na ordem inversa em que foram criados (primeiro lê o último contentor criado).

Dada uma instância com itens de pesos 82, 43, 40, 15, 12, 6 e contentores com capacidade igual a 100. A heurística LFD irá alocar o item com peso 82 ao primeiro contentor, o de peso 43 no segundo contentor, o de peso 40 no segundo contentor, o de peso 15 no segundo contentor, o de peso 12 no primeiro contentor e o último item de peso 6 no primeiro contentor, gerando uma solução ótima com dois contentores. O pseudocódigo da heurística LFD é apresentado na Figura 9.

```

1: procedimento LFD ( $N, w_i, L$ )
2:  $K = 0$ 
3: Para cada item  $i \in L$ 
4:     Para cada contentor  $j \in K$ 
5:          $l_j = \text{nível\_do\_bin}(j)$ 
6:         Se  $(l_j + w_i \leq C)$  Então
7:              $l_j += w_i$ 
8:              $S_H(j) = S_H(j) \cup \{i\}$ 
9:         Pare
10:        Senão
11:            Se  $(j = 1)$  então
12:                 $K += 1$ 
13:                 $S_H(K) = \{i\}$ 
14:            Fim se
15:        Fim se
16:         $j -= 1$ 
17:    Fim Para
18: Fim Para
18: Saída ( $K, S_H$ )

```

Figura 9 – Pseudocódigo da heurística Last Fit Decreasing – LFD

2.4.7 Heurística *Best Fit Decreasing* - BFD

A heurística *Best Fit Decreasing* – BFD é uma das mais utilizadas na literatura de problemas de empacotamento unidimensional.

Consiste em ordenar a lista de itens em ordem decrescente de peso e alocá-los ao primeiro contentor que ficar com a maior capacidade ocupada após a alocação do item desde que este não exceda a capacidade C do contentor.

Se não houver nenhum contentor que atenda a restrição, abre-se um novo contentor

O pseudocódigo da heurística BFD é descrito na Figura 10.

Dado um conjunto de elementos de pesos 82, 43, 40, 15, 12, 6 e contentores com capacidade 100. A heurística BFD irá alocar o item de peso 82 ao primeiro contentor, o de peso 43 ao segundo contentor, o de peso 40 ao segundo contentor, o de peso 15 ao segundo contentor (pois ele é o mais ocupado), 12 ao primeiro contentor e 6 ao primeiro contentor, gerando assim uma solução ótima para esta instância (Korf, 2002).

```

1: procedimento BFD ( $N, w_i, L$ )
2:  $K = 0$ 
3:  $gap = C$ 
4: Para cada item  $i \in L$ 
5:     Para cada contentor  $j \in K$ 
6:          $l_j = \text{nível\_do\_bin}(j)$ 
7:         Se  $(l_j + w_i \leq C)$  e  $(C - l_j - w_i < gap)$  Então
8:              $gap = C - l_j - w_i$ 
9:              $k = j$ 
10:        Senão
11:            Se  $(j = K)$  então
12:                 $K += 1$ 
13:                 $k = K$ 
14:        Fim se
15:    Fim se
16:    Fim Para
17:     $gap = C$ 
18:     $l_k += w_i$ 
19:     $S_H(k) = S_H(k) \cup \{i\}$ 
20:Fim Para
21:Saída ( $K, S_H$ )

```

Figura 10 – Pseudocódigo da heurística *Best Fit Decreasing* – BFD

2.4.8 Heurística *Worst Fit Decreasing* - WFD

A heurística *Worst Fit Decreasing* - WFD consiste em ordenar a lista de itens em ordem decrescente de peso e aloca-los ao primeiro contentor que ficar com a maior capacidade ociosa após a alocação do item desde que este não exceda a capacidade C do contentor. Um novo contentor será aberto se não houver nenhum que atenda a restrição de capacidade.

Usando a instância exemplo de Korf (2002), um conjunto de elementos de pesos 82, 43, 40, 15, 12, 6 e contentores com capacidade 100. A heurística WFD irá alocar o item de peso 82 ao primeiro contentor, o de peso 43 ao segundo contentor, o de peso 40 ao segundo contentor, o de peso 15 ao primeiro contentor (pois ele é o menos ocupado), 12 ao segundo contentor e 6 ao terceiro contentor, gerando assim uma solução com três contentores.

A heurística *Worst Fit Decreasing* – WFD é detalhada no pseudocódigo da Figura 11.

```

1: procedimento WFD ( $N, w_i, L$ )
2:  $K = 0$ 
3:  $gap = 0$ 
4: Para cada item  $i \in L$ 
5:     Para cada contentor  $j \in K$ 
6:          $l_j = \text{nível\_do\_bin}(j)$ 
7:         Se  $(l_j + w_i \leq C)$  e  $(C - l_j - w_i > gap)$  Então
8:              $gap = C - l_j - w_i$ 
9:              $k = j$ 
10:        Senão
11:            Se  $(j = K)$  então
12:                 $K += 1$ 
13:                 $k = K$ 
14:        Fim se
15:    Fim se
16:    Fim Para
17:     $gap = 0$ 
18:     $l_k += w_i$ 
19:     $S_H(k) = S_H(k) \cup \{i\}$ 
20: Fim Para
21: Saída ( $K, S_H$ )

```

Figura 11 – Pseudocódigo da heurística *Worst Fit Decreasing* – WFD

2.4.9 Heurística *Next Fit Decreasing* - NFD

Outra heurística bastante popular, a heurística *Next Fit* – NF, pode ser vista como um método mais intuitivo, alocando o maior número de itens ao contentor corrente enquanto o espaço disponível seja maior ou igual que o tamanho do item considerado. Quando a lista de itens está ordenada em ordem decrescente de peso, o algoritmo é chamado *Next Fit Decreasing* – NFD e consiste em alocar os itens ao último contentor criado sempre que houver disponibilidade de capacidade. Caso não exista capacidade disponível, fecha o contentor corrente, abre um novo contentor e aloca o item nele.

Usando a instância exemplo de Korf (2002), um conjunto de elementos de pesos 82, 43, 40, 15, 12, 6 e contentores com capacidade 100. A heurística NFD irá alocar o item de peso 82 ao primeiro contentor, o de peso 43 ao segundo contentor, o de peso 40 ao segundo contentor, o de peso 15 ao segundo contentor, o item de peso 12 ao terceiro contentor e 6 ao terceiro contentor, gerando uma solução com três contentores. O pseudocódigo da heurística é apresentado na Figura 12.

```

1: procedimento NFD ( $N, w_i, L$ )
2:  $K = 1$ 
3:  $j = 1$ 
4: Para cada item  $i \in L$ 
5:      $l_j = \text{nível\_do\_bin}(j)$ 
6:     Se ( $l_j + w_i > C$ ) Então
7:          $j += 1$ 
8:          $K += 1$ 
9:     Fim se
10:     $l_j += w_i$ 
11:     $S_H(j) = S_H(j) \cup \{i\}$ 
12:Fim Para
13:Saída ( $K, S_H$ )

```

Figura 12 – Pseudocódigo da heurística *Next Fit Decreasing* – NFD

2.4.10 Heurística *Worst Solution* - WS

A heurística *Worst Solution* – WS proposta nesta pesquisa, consiste em alocar apenas um único item a cada contentor, de modo a ter a pior solução possível. Foi proposta para ser uma solução inicial “sem vícios”, sem características quaisquer, de modo que a verificar os resultados das heurísticas de melhoria e metaheurísticas sobre resultados iniciais muito ruins. Sendo esta uma das contribuições desta pesquisa.

O pseudocódigo da heurística *Worst Solution* - WS é apresentado na Figura 13.

```

1: procedimento WS( $N, w_i, L$ )
2:  $K = 0$ 
3:  $j = 1$ 
4: Para cada item  $i \in L$ 
5:     $l_j = w_i$ 
6:     $j += 1$ 
7:     $K += 1$ 
8:     $S_H(j) = S_H(j) \cup \{i\}$ 
9:Fim Para
10:Saída ( $K, S_H$ )

```

Figura 13 – Pseudocódigo da heurística *Worst Solution* – WS

3 MODELAGEM MATEMÁTICA E MÉTODOS DE SOLUÇÃO

Nesta seção está descrita a modelagem matemática do problema de empacotamento unidimensional tratado nesta pesquisa e uma descrição dos métodos propostos para solucionar o problema.

3.1 MODELAGEM MATEMÁTICA

Esta seção descreve o modelo matemático para os problemas de empacotamento unidimensional, e uma função objetivo alternativa que será utilizada no âmbito dos métodos desenvolvidos.

3.1.1 Modelo matemático do Problema de Empacotamento Unidimensional

Dado um conjunto N de itens com pesos unitários $w_j, j = 1, 2, 3, \dots, N$, e um conjunto limitado de K contentores, $k = 1, 2, 3, \dots, K$ todos com capacidade C , o problema de empacotamento unidimensional consiste em determinar o mínimo número m de contentores, capaz de alocar todos os itens satisfazendo a restrição de ocupação máxima da capacidade dos contentores. Sendo y_k uma variável binária de abertura do contentor k e x_{jk} uma variável binária de alocação do item j ao contentor k , o modelo matemático pode ser enunciado da seguinte maneira:

$$\min m = \sum_{k=1}^K y_k \quad (15)$$

s.a.

$$\sum_{k=1}^K x_{jk} = 1, \quad \forall j \ j = 1, 2, 3, \dots, N \quad (16)$$

$$\sum_{j=1}^N w_j x_{jk} \leq C y_k, \quad \forall k \ k = 1, 2, 3, \dots, K \quad (17)$$

$$y_k \in \{0, 1\}, \quad \forall k \ k = 1, 2, 3, \dots, K \quad (18)$$

$$x_{jk} \in \{0, 1\}, \quad \forall k \ k = 1, 2, 3, \dots, K, \quad \forall j \ j = 1, 2, 3, \dots, N \quad (19)$$

A função objetivo (15) consiste em determinar o menor número de contentores m .

As restrições do modelo estabelecem:

- (16) determina que todo item j deve estar alocado a um único contentor k .
- (17) estabelece que a capacidade máxima C do contentor k deve ser respeitada, desde que este seja utilizado.
- (18) impõe o espaço de solução das variáveis binárias y_k de abertura dos contentores.
- (19) indica o espaço de solução das variáveis binárias x_{jk} de alocação do item j ao contentor k .

A função objetivo explicita o número de contentores, cuja soma deve ser minimizada. Mas dificulta a avaliação da ocupação dos contentores. A busca pela solução ótima envolve incrementar a ocupação dos contentores, pois isso pode possibilitar a redução do número de contentores necessários. Por este motivo, a função objetivo foi reformulada para expressar, de forma mais apropriada, a ocupação dos contentores, e permitir avaliar os ganhos de trocar os itens entre contentores.

O modelo pode ser generalizado para o caso de objetos diferentes, em que as capacidades dos contentores mudam, fazendo C_s , $s = 1, 2, 3, \dots, S$ na restrição (17).

3.1.2 Reformulação da Função Objetivo

A partir da constatação de que a função objetivo deveria permitir avaliar os ganhos resultantes da modificação de uma solução corrente em uma dada iteração de uma heurística de busca, e que, de forma intuitiva, uma boa solução é aquela cujos contentores têm alta ocupação (Fleszar e Hindi, 2002), optou-se por reescrever a função objetivo, a qual visa a maximização da ocupação dos contentores, em uma forma equivalente dada por:

$$G = \sum_{k=1}^{K_i} \left(\sum_{j=1}^{N_k} \frac{w_j x_{jk}}{C} \right)^p \quad (20)$$

Onde:

- K_i é o número de contentores da solução na iteração i da heurística de busca.

- N_k é o número de itens alocados no contentor k .
- C é a capacidade do contentor k .
- w_j é o peso do item j .
- p é um número inteiro, $p \geq 2$.
- x_{jk} são variáveis binárias de alocação do item j ao contentor k .

A função objetivo definida pela equação (20) depende da capacidade C dos contentores da instância utilizada no teste.

Ao introduzirmos um fator $1/L_{BPP}$ na definição da função objetivo (20), esta fica melhor quanto mais próximo for o valor de 1, ou quão próximo de L_{BPP} é a quantidade corrente de contentores, permitindo desta forma a comparação do valor da função objetivo de forma independente da instância utilizada.

$$G = \frac{1}{L_{BPP}} \sum_{k=1}^{K_i} \left(\sum_{j=1}^{N_k} \frac{w_j x_{jk}}{C} \right)^p \quad (21)$$

Onde L_{BPP} é o melhor limitante inferior calculado para o problema dado pela equação (13).

Dado um contentor k , o seu nível é definido como:

$$l_k = \sum_{j=1}^{N_k} w_{jk} \quad (22)$$

Onde w_{jk} é o peso do item j alocado ao contentor k e N_k é o número de itens alocados ao contentor k .

A variação na função objetivo pode ser calculada da mesma forma que a implementada por Fleszar e Hindi (2002) e Loh et al (2008).

Dados um item γ de um contentor α de nível l_α , um item δ de um contentor β de nível l_β e um número inteiro $p \geq 2$, as trocas entre estes itens produzem uma variação na função objetivo (21) dada pela expressão (23), observando que as ocupações dos demais contentores se cancelam no cálculo:

$$\Delta G = \frac{1}{L_{BPP} C^p} \{ [l_\alpha - w_\gamma + w_\delta]^p + [l_\beta + w_\gamma - w_\delta]^p - l_\alpha^p - l_\beta^p \} \quad (23)$$

Se definir a diferença entre os pesos trocados $\Delta w \equiv w_\gamma - w_\delta$, a equação (23) pode ser reescrita como:

$$\Delta G = \frac{1}{L_{BPP} C^p} \{ [l_\alpha - \Delta w]^p + [l_\beta + \Delta w]^p - l_\alpha^p - l_\beta^p \} \quad (24)$$

Para reduzir o número de contentores com as trocas entre itens faz-se necessário maximizar a função objetivo G , o que equivale a ter com a troca dos itens a cada iteração i :

$$\Delta G = G_i - G_{i-1} > 0 \quad (25)$$

Adotando-se o parâmetro $p = 2$:

$$\Delta G = \frac{1}{L_{BPP} C^2} \{ [l_\alpha - \Delta w]^2 + [l_\beta + \Delta w]^2 - l_\alpha^2 - l_\beta^2 \} = \frac{2\Delta w}{L_{BPP} C^2} \{ \Delta w - l_\alpha + l_\beta \} \quad (26)$$

Para que ΔG nessa forma atenda a relação (25) é necessário que $\Delta w - l_\alpha + l_\beta > 0$ e $\Delta w > 0$, ou $\Delta w - l_\alpha + l_\beta < 0$ e $\Delta w < 0$, logo:

$$w_\gamma > w_\delta \text{ e } w_\gamma > l_\alpha - l_\beta + w_\delta \quad (27) \text{ ou}$$

$$w_\gamma < w_\delta \text{ e } w_\gamma < l_\alpha - l_\beta + w_\delta \quad (28)$$

De forma semelhante, adotando-se $p = 3$:

$$\begin{aligned} \Delta G &= \frac{1}{L_{BPP} C^3} \{ [l_\alpha - \Delta w]^3 + [l_\beta + \Delta w]^3 - l_\alpha^3 - l_\beta^3 \} = \frac{3\Delta w}{L_{BPP} C^3} \{ l_\beta^2 - l_\alpha^2 + \Delta w(l_\alpha + l_\beta) \} \\ &= \frac{3\Delta w}{L_{BPP} C^3} \{ (l_\alpha - l_\beta)(l_\alpha + l_\beta) + \Delta w(l_\alpha + l_\beta) \} \\ &= \frac{3(l_\alpha + l_\beta)\Delta w}{L_{BPP} C^3} \{ l_\alpha - l_\beta + \Delta w \} \quad (29) \end{aligned}$$

Para que ΔG nessa forma atenda a relação (25) é necessário que $\Delta w - l_\alpha + l_\beta > 0$ e $\Delta w > 0$, ou $\Delta w - l_\alpha + l_\beta < 0$ e $\Delta w < 0$, obtendo novamente as relações (27) e (28).

Se for adotado $p = 4$:

$$\begin{aligned} \Delta G &= \frac{1}{L_{BPP} C^4} \{ [l_\alpha - \Delta w]^4 + [l_\beta + \Delta w]^4 - l_\alpha^4 - l_\beta^4 \} \\ &= \frac{2\Delta w}{L_{BPP} C^3} \{ 2(l_\beta^3 - l_\alpha^3) + 3\Delta w(l_\beta^2 + l_\alpha^2) + 2\Delta w^2(l_\beta - l_\alpha) + \Delta w^3 \} \quad (30) \end{aligned}$$

Para determinar os sinais dessa inequação é preciso encontrar as raízes de um polinômio de grau 3, o que pode ser resolvido numericamente e consome mais tempo de processamento.

Devido aos tempos de processamento necessários para a obtenção das raízes do polinômio (30) e aos resultados obtidos por Falkenauer (1996) que indicam que para $p > 2$ a convergência do algoritmo é prematura, optou-se neste estudo pela utilização do parâmetro $p = 2$ para produzir diversificação e intensificação na busca.

As relações (27) e (28) são válidas também quando um único item é retirado de contentor α e colocado em outro contentor β (movimento de inserção), que é o movimento que permite zerar a capacidade ocupada e elimina um contentor α da solução. Para isso basta fazer $w_\delta = 0$ na relação (27). Um exemplo de movimento de inserção é ilustrado na Figura 14.

Figura 14 – Exemplo de movimento de inserção de um item γ em um contentor β

A validade das relações (27) e (28) também ocorre quando um grupo de itens de um contentor é trocado por um grupo de itens de outro contentor. Bastando, para isso, substituir w_γ e w_δ pela soma dos pesos dos itens movimentados nas relações (27) e (28). Assim a validade das relações ocorre em todos os movimentos que melhoram a função objetivo e consequentemente reduzem a quantidade utilizada de contentores.

Um exemplo de movimento de troca de itens entre contentores que aumentam a ocupação de um dos contentores é ilustrado na Figura 15.

Figura 15 – Exemplo de movimento de troca de itens entre contentores

Figura 16 – Exemplo de movimento de troca do tipo um item por dois itens

Em alguns artigos da literatura de empacotamento unidimensional com suas variantes, pode ser encontrada uma das relações de melhoria, a relação matemática (27) e o caso particular dela no movimento de inserção com $w_\delta = 0$ como no trabalho de Dell'Amico et al (2012), porém sem uma justificativa para a sua utilização ou demonstração de sua validade. Desta forma em seu trabalho, Dell'Amico et al (2012) podem ter deixado de encontrar melhorias na função objetivo por não terem utilizado a segunda relação de melhoria definida pela inequação (28), o que melhoria

seus resultados por encontrar mais soluções melhores e poder reduzir seus tempos de processamento.

Nos algoritmos propostos nesta pesquisa para resolver os problemas de empacotamento foram utilizados quatro tipos de movimentos (Fleszar e Hindi, 2002):

- Movimento de inserção: retirada de um item γ de um contentor α e inserção em um contentor β , conforme Figura 14.
- Movimento de troca do tipo um item por um item: Troca de itens entre contentores, conforme Figura 15.
- Movimento de troca do tipo um item por dois itens: Troca de um par de itens com um único item entre contentores, conforme Figura 16.
- Movimento de troca do tipo dois itens por dois itens: Troca de dois pares de itens entre contentores, conforme Figura 17.

Figura 17– Exemplo de movimento de troca do tipo dois itens por dois itens

3.2 MÉTODOS DE SOLUÇÃO

Esta seção descreve os métodos de solução para problemas de empacotamento unidimensional propostos neste trabalho.

São propostas a heurística diferença de quadrados que busca melhorias em soluções iniciais geradas por outras heurísticas e a metaheurística *simulated annealing* implementada para gerar aleatoriedade e melhorias nas soluções.

3.2.1 Heurística Diferença de Quadrados

Na Figura 18 é descrito o pseudocódigo da heurística Diferença de Quadrados (Difference-of-Squares – DS) proposta como uma das principais contribuições desta pesquisa.

O procedimento começa com a leitura da solução inicial por uma heurística S_H , o número máximo de iterações pré-determinado da busca $iter_{max}$, o melhor limitante inferior calculado para o problema de empacotamento unidimensional L_{BPP} , a capacidade C , os pesos w_j e lista de itens da instância L .

O número K recebe a quantidade de contentores da solução inicial da heurística S_H , a solução do procedimento S_i recebe a solução inicial da heurística S_H , a lista de itens candidatos a serem transferidos de um contentor para o outro L^* recebe a lista L com todos os itens da instância, a lista de itens alterados na iteração anterior L' é esvaziada e o contador de iterações $iter$ é iniciado com 1.

Em seguida, o procedimento entra em loop testando as trocas de itens entre contentores que geram melhorias na função *fitness*. Assim, o peso de cada item da lista L de todos os itens é comparado ao peso de cada um dos outros itens. Um contador L' que registra a quantidade de itens alterados na iteração anterior é iniciado com a quantidade total de itens da instância e a cada varredura da lista L o contador é reiniciado em zero.

Os movimentos testados nessa parte do algoritmo são dos tipos inserção de um item de um contentor em outro e troca entre contentores de um item por outro item.

Os critérios de parada da busca são:

- Número de contentores K da solução igual ao melhor limitante inferior L_{BPP} .
- Número máximo de iterações atingido $iter_{max}$.
- Contador de itens movimentados na iteração anterior L' nulo.

Para cada item γ da lista com todos os itens e para cada item δ da lista de todos os itens da instância, o procedimento compara as diferenças de pesos entre os itens e

de níveis dos contentores onde eles estão alocados, verificando quais delas atendem as relações de melhoria na função objetivo e realizam os movimentos de inserção de um item de um contentor em outro e inter-troca de itens entre contentores sempre que as relações de melhoria são observadas. Quando um dos critérios de parada é atingido a busca retorna o número de contentores K e a melhor solução S_i obtida na busca.

```

1: Entrada ( $S_H, iter_{max}, L_{BPP}, C, w_j, L$ )
2:  $K = \text{Número\_de\_bins}(S_H)$ 
3:  $S_i = S_H$ 
4:  $L^* = L$ 
5:  $iter = 1$ 
6:  $L' = L$ 
7: Enquanto ( $iter \leq iter_{max}, K > L_{BPP}, L' \neq 0$ )
8:    $L' = 0$ 
9:   Para cada item  $\gamma \in L$ 
10:      $\alpha = \text{bin\_do\_item}(\gamma)$ 
11:      $l_\alpha = \text{nível\_do\_bin}(\alpha)$ 
12:     Para cada item  $\delta \in L^*$ 
13:        $\beta = \text{bin\_do\_item}(\delta)$ 
14:        $l_\beta = \text{nível\_do\_bin}(\beta)$ 
15:       Se ( $w_\gamma > l_\alpha - l_\beta$  e  $l_\beta + w_\gamma \leq C$ ) Então
16:          $L' = L' + 1$ 
17:         Insere item  $\gamma$  no contentor  $\beta$ 
18:       Senão
19:         Se ( $w_\gamma - w_\delta > l_\alpha - l_\beta$  e  $w_\gamma > w_\delta$  e
20:            $l_\alpha + w_\delta - w_\gamma \leq C$  e  $l_\beta + w_\gamma - w_\delta \leq C$ ) ou
21:           ( $w_\gamma - w_\delta < l_\alpha - l_\beta$  e  $w_\gamma < w_\delta$  e
22:              $l_\alpha + w_\delta - w_\gamma \leq C$  e  $l_\beta + w_\gamma - w_\delta \leq C$ ) Então
23:                $L' = L' + 2$ 
24:               Swap itens  $\gamma$  e  $\delta$  nos contentores  $\alpha$  e  $\beta$ 
25:         Fim se
26:       Fim se
27:     Fim Para
28:   Fim Para
29:    $K = \text{Número\_de\_bins}(S_i)$ 
30:    $iter = iter + 1$ 
31:    $L^* = L'$ 
32: Fim Enquanto
33: Saída ( $K, S_i$ )

```

Figura 18 – Pseudocódigo da heurística diferença de quadrados

3.2.2 Exemplo da heurística Diferença de Quadrados

Nesta seção é descrito um exemplo detalhado da heurística diferença de quadrados – Difference-of-Squares – DS, proposta na pesquisa. Para ilustrar o seu funcionamento é utilizada uma instância do conjunto ffd-hardD da literatura dos problemas BPP. A instância utilizada foi a BPP16_iD4_20 proposta no artigo de Alvim *et al* (2004). Essa é uma instância classificada como FFD hard, ou seja, uma instância para o qual o algoritmo FFD não gera uma solução ótima.

A instância BPP16_iD4_20 utilizada para exemplo possui contentores de capacidade 1000 e 20 itens com pesos 398, 397, 389, 388, 383, 372, 363, 361, 338, 318, 310, 301, 299, 286, 284, 280, 270, 270, 265 e 264, conforme ilustrado na Figura 19.

Figura 19 – Instância utilizada no exemplo da heurística diferença de quadrados

Figura 20 – Exemplo de solução inicial gerada pela heurística FFD

O procedimento começa com a geração de uma solução inicial viável por uma heurística simples como a FFD, conforme ilustrado na Figura 20. Em seguida o algoritmo testa os movimentos de inserção de cada item em cada um dos contentores e de trocas de contentores entre cada um dos pares de itens. No exemplo o primeiro movimento testado é a inserção do item 1 de peso 398 no contentor 2. O movimento não é executado, por ser inviável, pois excede a capacidade do contentor, conforme ilustrado na Figura 21.

O passo seguinte consiste no teste da inter-troca entre contentores do item 1 com o item 2 de pesos respectivamente 398 e 397. Como ambos estão no mesmo contentor, a troca não é executada.

O teste seguinte é a inter-troca entre contentores do item 1 com o item 3 (de pesos 398 e 389 respectivamente). A troca é viável, mas não atende às relações de melhoria da função objetivo (27) e (28), gerando redução na função objetivo modificada (função *fitness*), conforme Figura 22. Por isso o movimento não é executado.

Figura 21 – Exemplo de movimento de inserção inviável.

Em seguida a inserção do item 1 em cada um dos outros contentores e troca do item 1 com cada um dos outros itens é testada, buscando um movimento viável que atenda as relações de melhoria da função *fitness* (27) e (28).

Durante os testes dos movimentos, verifica-se que o primeiro movimento de troca que é viável e que melhora a função *fitness* é a troca de contentores entre o item 1 de peso 398 e o item 12 de peso 301, conforme Figura 23.

O teste seguinte consiste nos movimentos de inserção do item 2 em cada um dos contentores e troca entre contentores do item 2 com cada um dos outros itens. Os resultados indicam que a troca entre contentores dos itens 2 e 3 é viável e resulta em melhoria na função *fitness*, conforme Figura 24.

Figura 22 – Exemplo de movimento de troca com redução na função objetivo

Figura 23 – Troca entre contentores do item 1 com o item 12

Figura 24 – Troca entre contentores dos itens 2 e 3 com melhoria na função objetivo

Figura 25 – Movimento de inserção viável do item 19 no contentor 4

Dando sequência no procedimento, o algoritmo identifica que as trocas entre os contentores dos itens 3 e 4; em seguida entre os itens 4 e 5; entre os itens 5 e 6; entre os itens 6 e 9; inserção do item 8 no contentor 1; troca entre contentores dos itens 15 e 2; inserção do item 16 no contentor 2; inserção do item 18 no contentor 7; inserção do item 19 no contentor 4 (conforme Figura 25); inserção do item 20 no contentor 4, que esvazia o contentor 8, reduzindo o número de contentores utilizados e os igualando ao número de contentores do melhor limitante inferior L_{BPP} , comprovando desta forma que uma solução ótima foi obtida pelo algoritmo, conforme Figura 26.

Figura 26 – Redução no número de contentores com a heurística diferença de quadrados

3.2.3 Metaheurística *simulated annealing*

A metaheurística *simulated annealing* - SA foi desenvolvida por Kirkpatrick em 1983 para otimização de atividades econômicas fundamentada em um processo análogo ao da termodinâmica de recozimento em metalurgia. No processo físico um material é aquecido a altas temperaturas, e em seguida resfriado lentamente até a solidificação de forma controlada, fazendo com que os átomos do material se organizem numa estrutura uniforme de modo a alcançar um estado de energia

mínima, reduzindo a possibilidade de defeitos no material. Na metaheurística um parâmetro T análogo à temperatura do processo físico, controla a probabilidade de que soluções vizinhas ruins substituam a solução incumbente.

A cada iteração da busca é sorteado um número aleatório η no intervalo $[0,1[$ que é comparado a uma probabilidade $P(\Delta G)$ utilizando o critério de Metropolis e a distribuição de Boltzmann para a aceitação de novas soluções conforme a função:

$$P(\Delta G) = \begin{cases} 1 & \text{se } \Delta G > 0 \\ e^{\left(\frac{\Delta G}{T}\right)} & \text{caso o contrário} \end{cases} \quad (31)$$

Onde ΔG é a diferença na função objetivo entre a solução incumbente e uma solução vizinha, sendo análoga a variação da energia no processo físico.

Nos problemas de maximização há três possibilidades de acordo com os valores de ΔG :

- $\Delta G > 0$ – Há melhoria na função objetivo, pois os contentores mais ocupados tendem a ter uma ocupação maior e os menos ocupados tendem a ficar com uma ocupação menor, aumentando a possibilidade de ficarem vazios.
- $\Delta G = 0$ – Não há alteração na função objetivo.
- $\Delta G < 0$ – Esse tipo de variação piora o objetivo.

O algoritmo de SA é iniciado para melhorar a solução inicial obtida por uma das heurísticas descritas no capítulo 2 (FFD, LFD, BFD, WFD, NFD, MBS, MBS', MBSR, SSD e WS) caso o melhor dos limitantes inferiores descritos no capítulo 2 não seja atingido por elas.

O procedimento começa com a leitura do parâmetro de resfriamento da temperatura σ , número de itens da instância N , lista de pesos dos itens L , número de iterações a temperatura constante conhecido na literatura como *epoch length* ou Markov *chain length* M_C , capacidade dos contentores C e o melhor limitante inferior obtido L_{BPP} .

Na primeira parte do procedimento, a melhor solução S_B , a solução incumbente S_i e a solução vizinha S_v recebem a solução inicial de uma heurística S_H .

A temperatura inicial T_i é calculada como uma função do número de itens N e do limitante inferior L_{BPP} . Outra função calcula a temperatura final da busca T_F com os mesmos parâmetros. O número de contentores K recebe o número de contentores da solução inicial.

Para cada faixa de temperatura é executado um procedimento que perturba a solução incumbente gerando uma solução vizinha S_V durante um número de iterações pré-determinado M_C , conhecido como *Markov Chain Length*, que pode ser um parâmetro inicial do algoritmo, ou calculado de forma dinâmica como no artigo de Atiqullah (2004) para assegurar equilíbrio a baixas temperaturas. As soluções vizinhas S_V são geradas por movimentos viáveis aleatórios dos tipos inserção, inter-troca, troca de um item com um par de itens e troca entre dois pares de itens descritos na seção 3.2.3. Movimentos inviáveis também são aceitos, mas os itens sobressalentes dos contentores são retirados do contentor e reinseridos na solução através do algoritmo *First Fit* – FF, tornando a solução viável novamente.

A cada iteração é sorteado um valor η entre zero e um que é comparado ao valor de $\exp(\Delta G_i/T)$. Onde ΔG_i é a taxa de variação da função objetivo G . Sempre que $\Delta G_i > 0$ ou $\eta < \exp(\Delta G_i/T)$, a solução incumbente é atualizada pela solução vizinha S_V e sempre que $\Delta G_i > 0$ a melhor solução S_B é atualizada pela solução incumbente S_i . Após um número M_C de iterações o procedimento reduz a temperatura por um fator σ e o processo irá se repetir enquanto $T > T_F$, $K > L_{BPP}$ e o número máximo de iterações sem melhorias não é atingido. Ao final do processamento o algoritmo retorna a melhor solução encontrada S_B .

A parametrização da busca tem grande influência nos resultados, por isso são necessárias boas estimativas para os parâmetros da busca. Por esse motivo foram propostas duas agendas de parâmetros (conhecidas na literatura por *annealing schedule*).

Na primeira agenda foram utilizadas as estimativas propostas por Rao e Iyengar (1993) para a taxa de redução de temperatura e para os parâmetros de temperatura inicial e final, combinadas com algumas modificações propostas nesta pesquisa.

A temperatura inicial deve ser alta o suficiente para permitir que soluções piores sejam aceitas, mas não tão alta para que muitas soluções desse tipo sejam aceitas no algoritmo (Brusco et al, 1997). Por isso neste trabalho os autores definiram as temperaturas iniciais com o percentual de movimentos aceitáveis no início do algoritmo.

A temperaturas altas os movimentos para gerar soluções vizinhas serão aceitos somente se

$$e^{(\Delta G_i/T_i)} \leq \frac{1}{N} \quad (32)$$

Assim apenas a movimentação de um entre os N itens seria aceita, conforme Rao e Iyengar (1993). Então:

$$\frac{\Delta G}{T_i} \approx \ln\left(\frac{1}{N}\right) \Rightarrow \frac{\Delta G}{T_i} \approx \ln(N^{-1}) \Rightarrow \frac{\Delta G}{T_i} \approx -\ln(N) \Rightarrow T_i \approx -\frac{\Delta G}{\ln(N)} \quad (33)$$

Substituindo o valor de ΔG nos movimentos tem-se:

$$T_i \approx \left| \frac{2(w_\gamma - w_\delta)}{L_{BPP} C^2 \ln(N)} \{w_\gamma - w_\delta - l_\alpha + l_\beta\} \right| \quad (34)$$

Como a função objetivo desta pesquisa é diferente da utilizada por Rao e Iyengar (1993), a expressão da estimativa da temperatura também fica diferente da obtida pelos autores, sendo esta outra contribuição desta pesquisa.

Para que a variação ΔG tenha um valor alto o que ocorre, por exemplo, com $w_\delta = 0$ e $w_\gamma = w_{max} \equiv \max\{w_j: j \in N\}$.

Fazendo uma aproximação de $l_\alpha \approx l_\beta$ e substituindo os valores encontramos uma estimativa para a temperatura inicial:

$$T_i \approx \frac{2w_{max}^2}{L_{BPP} C^2 \ln(N)} \quad (35)$$

A menor variação ΔG da função objetivo revisada pode ser estimada facilmente (Rao e Iyengar, 1993). Considerando o item de menor peso $w_{min} \equiv \min\{w_j: j \in N\}$, temos que apenas a movimentação de um entre os N itens será aceita (Rao e Iyengar, 1993). Fazendo novamente uma aproximação de $l_\alpha \approx l_\beta$ tem-se que:

$$T_F \approx \frac{2w_{min}^2}{L_{BPP} C^2 \ln(N)} \quad (36)$$

Como a temperatura inicial é calculada como uma função do número de itens N e do limitante inferior L_{BPP} , outra função calcula a temperatura final da busca T_F com os mesmos parâmetros e a temperatura T é uma função da temperatura inicial. Assim, é possível fazer uma analogia com o processo físico da termodinâmica onde aparece a constante de Boltzman K_B , verificando que para cada instância aparece uma constante que só depende das características do problema (limitante inferior e capacidade dos contentores):

$$K_B = 1/(L_{BPP}C^2) \quad (37)$$

Gerando desta forma respectivamente as estimativas da temperatura inicial e final do algoritmo:

$$T_\infty = \frac{2w_{max}^2}{\ln(N)} \quad (38)$$

$$T_0 = \frac{2w_{min}^2}{\ln(N)} \quad (39)$$

E modificando os parâmetros de temperatura com a inclusão de uma constante de *bin packing* K_B no *simulated annealing*.

Na segunda agenda de parâmetros, nas temperaturas altas os movimentos para gerar soluções vizinhas serão aceitos uma quantidade maior de movimentos, em vez de um movimento em N como na primeira agenda, foi considerado um movimento em 3. Para simplificar a expressão da temperatura, foi utilizado e no lugar de 3. Aceitando o movimento somente se;

$$e^{(\Delta G_i/T_i)} \leq \frac{1}{e} \quad (40)$$

O que elimina o logaritmo da expressão. Então:

$$\frac{\Delta G}{T_i} \approx \ln\left(\frac{1}{e}\right) \Rightarrow \frac{\Delta G}{T_i} \approx \ln(e^{-1}) \Rightarrow \frac{\Delta G}{T_i} \approx -\ln(e) \Rightarrow T_i \approx -\Delta G \quad (41)$$

Substituindo o valor de ΔG nos movimentos tem-se:

$$T_i \approx \left| \frac{2(w_\gamma - w_\delta)}{L_{BPP}C^2} \{w_\gamma - w_\delta - l_\alpha + l_\beta\} \right| \quad (42)$$

Para que a variação ΔG tenha um valor alto o que ocorre, por exemplo, com $w_\delta = 0$ e $w_\gamma = w_{max} \equiv \max\{w_j: j \in N\}$

Fazendo as aproximações de $l_\alpha \approx C$ e $l_\beta \approx w_{min}$, com a justificativa de que os contentores mais cheios estariam com aproximadamente a capacidade do contentor

completa e os mais vazios, com apenas um item, teriam o nível aproximadamente igual ao peso do item de menor peso, teríamos $w_{min} \equiv \min\{w_j: j \in N\}$ e substituindo os valores encontramos uma estimativa para a temperatura inicial:

$$T_i \approx \frac{|2w_{max}(w_{max} - C + w_{min})|}{L_{BPP}C^2} \quad (43)$$

Para a temperatura final foi utilizada a mesma estimativa da primeira agenda.

Gerando desta forma respectivamente as estimativas da temperatura inicial e final do algoritmo:

$$T_{\infty} = |2w_{max}(w_{max} - C + w_{min})| \quad (44)$$

$$T_0 = \frac{2w_{min}^2}{\ln(N)} \quad (45)$$

Conforme exemplificado no Anexo A, estas aproximações permitem obter gráficos de probabilidade de aceitação de soluções piores em função da temperatura mais próximos dos esperados na teoria do *simulated annealing*, sendo esta outra contribuição desta pesquisa.

O fator de resfriamento σ deve ser alto o suficiente para garantir que a probabilidade de aceitar soluções ruins no final do algoritmo seja baixa, porém não pode ser tão alto que reduza a temperatura muito rapidamente, interferindo na convergência do algoritmo (Brusco et al, 1997). O parâmetro Markov Chain Length M_C , que geralmente é um múltiplo pequeno do tamanho do problema, pode ser estimado com o número máximo de iterações da busca e a relação entre as temperaturas final e inicial com o fator de resfriamento. Mas nesta pesquisa foi aproximado pelo número de itens da instância, permitindo que em média cada item tenha a mesma probabilidade de ser movido de um contentor para o outro em cada faixa de temperatura constante.

A Figura 28 apresenta o fluxograma do algoritmo proposto para resolução do BPP. A primeira parte do algoritmo consiste na leitura pelo programa em C++ dos dados das instâncias que estão armazenados em arquivos .txt. Em seguida são calculados os limitantes inferiores descritos na seção 2.3. Seguido do cálculo de cada um dos dez limitantes superiores descritos na seção 2.4.

Se o número de contentores do limitante superior for igual ao número de contentores do limitante inferior, uma solução ótima foi obtida, o algoritmo não precisa executar as heurísticas de melhoria da solução inicial e os resultados são exportados para um arquivo .txt.

```

1: procedimento  $SA(S_H, \sigma, M_C, N, L, C, L_{BPP}, i_{max})$ 
2:  $S_B = S_H$ 
3:  $S_i = S_H$ 
4:  $S_V = S_H$ 
5:  $T = T_{inicial}(N, L_{BPP})$ 
6:  $T_F = T_{final}(N, L_{BPP})$ 
7:  $K_B = 1/(L_{BPP}C^2)$ 
8:  $K = \text{Número\_de\_bins}(S_B)$ 
9:  $t = 0$ 
10: Enquanto  $(T > T_F, K > L_{BPP}, t \leq i_{max})$ 
11:      $i = 1$ 
12:     Enquanto  $(i \leq M_C)$ 
13:          $S_V = \text{Perturba}(S_i)$ 
14:          $\eta = \text{Randomico}()$ 
15:          $\Delta G_i = G(S_V) - G(S_B)$ 
16:         Se  $(\Delta G_i > 0)$  ou  $\left(\eta < \exp\left(\Delta G_i / K_B T\right)\right)$  Então
17:              $S_i = S_V$ 
18:         Fim se
19:         Se  $(\Delta G_i > 0)$  Então
20:              $S_B = S_V$ 
21:              $t = 0$ 
22:         Senão
23:              $t += 1$ 
24:         Fim se
25:          $i = i + 1$ 
26:     Fim Enquanto
27:      $T = \sigma T$ 
28: Fim Enquanto
27: Saída  $(S_B)$ 

```

Figura 27 – Pseudocódigo do *Simulated Annealing* - SA

Caso o número de contentores do limitante superior não seja igual ao número de contentores do limitante inferior, o algoritmo processa as combinações entre as heurísticas diferença de quadrados – DS e *simulated annealing* – SA (DS, SA, SA+DS, DS+SA e DS+SA+DS) aplicadas ao resultado da solução inicial, sendo estas contribuições desta pesquisa para a resolução dos problemas de empacotamento unidimensional.

No final o algoritmo faz um controle de qualidade das soluções obtidas para garantir que elas atendem todas as restrições do problema e uma planilha em Microsoft Excel lê e compara os resultados. Os resultados registrados na planilha Excel indicam a quantidade de soluções ótimas obtidas (limitante inferior igual ao limitante superior), percentual de instâncias com solução ótima obtida e as diferenças médias e máximas entre os limitantes inferior e superior, bem como o tempo de processamento das heurísticas.

Figura 28 – Fluxograma do programa para resolução dos BPP

4 TESTES COMPUTACIONAIS

Este capítulo descreve as principais instâncias disponíveis na literatura, as instâncias escolhidas para os testes, os testes computacionais realizados e os resultados obtidos com os métodos propostos.

As heurísticas foram codificadas em um programa em C++ com Visual Studio 2013® e os resultados ótimos para avaliação das heurísticas foram processados no programa computacional de otimização GUROBI® 6.5.2. As instâncias *benchmark* da literatura foram processadas em um computador HP Pavilion dm4 com processador Intel® Core™ i7 2620M 2.70 GHz com 8 GB RAM e 750 GB HDD.

4.1 INSTÂNCIAS PARA TESTE

Na literatura são propostas diversas instâncias para teste dos algoritmos. A ideia de compartilhar as instâncias para avaliar a qualidade dos algoritmos propostos surgiu nos anos 1990, inicialmente por e-mail (Beasley, 1990) e com o passar do tempo pela disponibilização nos sites dos grupos de pesquisa ou de pesquisadores independentes.

Alguns autores compartilham as instâncias através de algoritmos geradores de problemas. Gau e Wäsher (1995) apresentam um gerador de problemas de corte e empacotamento unidimensionais – (1D-CSP), classificado como 1/V/I/R pela tipologia de Dickhoff (1990), identificando os parâmetros e características para que as instâncias dos problemas tornem-se reproduzíveis. Um conjunto de instâncias de *benchmarking* é identificado para permitir comparação entre algoritmos propostos na literatura. O problema 1D-CSP é formalmente equivalente ao problema de 1D-BPP denotado por 1/V/I/M de acordo com a tipologia de Dickhoff (1990). No entanto em problemas reais do 1D-CSP diferentemente dos 1D-BPP uma propriedade característica pode ser observada. Enquanto que no problema típico de BPP as demandas são pequenas (próximas de um), nos problemas de corte e empacotamento as demandas são grandes (significativamente maiores que um),

conforme Gau e Wäsher (1995). Assim para os CSP um novo parâmetro precisa ser informado, a demanda média por peso dos itens.

Nos problemas de corte e empacotamento se observa que existe um *gap* muito pequeno entre o valor da solução ótima inteira em relação a solução do problema contínuo. Uma instância possui a propriedade IRUP – *Integer Round-Up Property* se este *gap* for menor que 1, caso o contrário, a instância é chamada de non-IRUP. Até a apresentação do artigo de Rietz et al (2002) poucas instâncias tinham sido publicadas possuindo *gaps* maiores do que 1, até que neste artigo os autores propuseram diversas famílias de instâncias com *gaps* maiores do que 1 e seu método de construção.

Os testes computacionais com instâncias geradas aleatoriamente mostraram que as instâncias non-IRUP são raramente geradas, mas são as mais difíceis para os métodos de solução exatos, conforme Rietz et al (2002), permitindo assim utilizá-las para testes de métodos de solução exatos e seu comportamento médio. Em geral as instâncias non-IRUP possuíam capacidades dos contentores da ordem de milhões e por isso acreditava-se que não ocorreriam em exemplos práticos, até a apresentação de um grande conjunto de instâncias non-IRUP com capacidades dos contentores da ordem de centenas (Caprara et al, 2015), mostrando que os *gaps* maiores que 1 podem ocorrer em exemplos de aplicações práticas. Os autores obtiveram milhares de instâncias com capacidades iguais a 100, 24 itens e pesos dos itens entre 22 e 61. A menor delas tem capacidade dos contentores igual a 146 e pesos entre 5 e 65. Devido à baixa quantidade de itens e capacidades dos contentores, os autores chamaram estas instâncias de “friendly” e as disponibilizaram na internet em seu site de resultados: http://www.or.unimore.it/resources/BPP_non_IRUP/instances.html.

Um dos conjuntos de instâncias mais difíceis utilizadas em artigos são as propostas por Falkenauer (1996), divididas em duas classes de 80 instâncias cada. Na primeira classe, chamada de Falkenauer U, os itens têm pesos distribuídos uniformemente, com número de itens por instância entre 120 e 1000 e capacidades dos contentores igual a 150. A segunda classe de instâncias, dita Falkenauer T ou “triplets” de

Falkenauer, agrupa três itens por contentor na solução ótima (um grande e dois pequenos) sem folga, e contém entre 60 e 501 itens por instância, com capacidades dos contentores iguais a 1000 e estão disponíveis para *download* no site de resultados do Grupo de Pesquisa Operacional de Universidade de Bologna (<http://or.dei.unibo.it/library/bpplib>). Todas as instâncias de Falkenauer (1996) têm solução ótima no limitante inferior, exceto uma conforme Gent (1998).

Segundo Falkenauer (1996) nem HGGA nem MTP (com limites de *backtrackings* imposto) conseguiam obter o limitante inferior em cinco instâncias, por isso Falkenauer (1996) conjecturou que estas instâncias não tinham solução ótima no limitante inferior. Gent (1998) apresentou solução para as cinco instâncias em aberto do conjunto de Falkenauer (1996), mostrando que em quatro destas instâncias o limitante inferior pode ser alcançado, resolvendo as instâncias até mesmo manualmente com algumas horas de cálculo ou poucos segundos de processamento em computador. Por isso o autor questiona a validade de qualquer conclusão a respeito de sua dificuldade.

Outra classe de instâncias muito utilizadas para teste e comparação de algoritmos é a classe proposta por Scholl et al (1997), que possui diversos níveis de dificuldade. As instâncias podem ser baixadas diretamente da página web com os resultados dos autores (<https://www2.wiwi.uni-jena.de/Entscheidung/binpp/index.htm>). Elas são divididas em três conjuntos com 720, 480 e 10 instâncias com quantidades de itens entre 50 e 500. As capacidades estão entre 100 e 150 (para o conjunto Scholl1), iguais a 100 (para o conjunto Scholl2) e iguais a 100.000 (para o conjunto Scholl3).

Uma classe de instâncias que estão entre as consideradas mais difíceis são as propostas por Wäscher et al (1996). Podem ser baixadas do site ESICUP do grupo de pesquisa operacional EURO - Special Interest Group on Cutting and Packing (<https://paginas.fe.up.pt/~esicup/datasets>). São 17 instâncias difíceis com quantidades de itens entre 27 e 239 e capacidade dos contentores iguais a 10.000.

As instâncias da classe Schwerin, utilizadas por Schwerin et al (1997), podem ser baixadas do site ESICUP (<https://paginas.fe.up.pt/~esicup/datasets>). Elas são

divididas em dois grupos, ditos Schwerin1 e Schwerin2, com 100 instâncias cada um e quantidade de itens de 100 e 120, respectivamente. A capacidade dos contentores é igual a 1000.

No relatório técnico de Schoenfield (2002) foram introduzidas 28 instâncias difíceis com quantidade de itens entre 160 e 200 e capacidade dos contentores iguais a 1000. Estas instâncias podem ser baixadas do site ESICUP <https://paginas.fe.up.pt/~esicup/datasets> e estão entre as mais utilizadas na literatura dos BPP.

Tabela 1 – Resumo dos dados das principais instâncias da literatura

Conjunto	Autores	Ano	# Inst.
AI	M. Delorme, M. Iori, and S. Martello	2016	250
ANI	M. Delorme, M. Iori, and S. Martello	2016	250
Caprara	Caprara et al	2015	4.955
Correia	Correia, Oliveira e Ferreira	2003	10
Cutgen	Gau, T. e Wäscher, G	1995	1.800
Falkenauer T	E. Falkenauer	1996	80
Falkenauer U	E. Falkenauer	1996	80
Hard28	J.E. Schoenfield	2002	28
Irnich	T. Gschwind and S. Irnich	2016	240
Liang	K. H. Liang, X. Yao, C. Newton, D. Hoffman	2002	10
Rand. Generated	M. Delorme, M. Iori e S. Martello	2016	3.840
Schoenfield	J. E. Schoenfield	2002	53
Scholl1	A. Scholl, R. Klein e C. Jürgens	1997	720
Scholl2	A. Scholl, R. Klein e C. Jürgens	1997	480
Scholl3	A. Scholl, R. Klein e C. Jürgens	1997	10
Schwerin1	P. Schwerin e G. Wäscher	1997	100
Schwerin2	P. Schwerin e G. Wäscher	1997	100
Wäscher Gau	G. Wäscher e T. Gau	1996	17
Total			13.023

Um conjunto variado de instâncias pode ser encontrado no trabalho de Delorme et al (2016). Elas formam um conjunto grande de instâncias, com quantidades de itens (50, 100, 200, 300, 400, 500, 750, 1 000), capacidades de contentores (50, 75, 100, 120, 125, 150, 200, 300, 400, 500, 750, 1 000), e pesos mínimos e máximos de itens variando entre $(0,1C; 0,2C)$ e $(0,7C; 0,8C)$ respectivamente. Para cada quadruplete 10 instâncias foram geradas formando um conjunto com 3.840 instâncias. Sendo estas consideradas fáceis. No mesmo trabalho de Delorme et al (2016) podem ser encontrados outros dois conjuntos de instâncias, conhecidas por Augmented Non IRUP e Augmented IRUP que também apresentam quantidades variadas de capacidades e itens por instância. Estas instâncias podem ser baixadas do site de instâncias do Grupo de Pesquisa Operacional de Universidade de Bologna (<http://or.dei.unibo.it/library/bpplib>).

Outro conjunto de difícil resolução foi proposto por Gschwind e Irnich (2016) e pode ser baixado do site do Grupo de Pesquisa Operacional de Universidade de Bologna (<http://or.dei.unibo.it/library/bpplib>). Um resumo dos dados das principais instâncias publicadas na literatura dos problemas de empacotamento é descrito na Tabela 1. São informados o nome do conjunto de instâncias, autores, ano de publicação e quantidade de instâncias do conjunto.

4.2 RESULTADOS DOS TESTES

Nesta seção são analisados os resultados dos testes realizados na pesquisa comparando cada um dos limitantes superiores obtidos com as heurísticas aos limitantes inferiores implementados, às melhores soluções conhecidas da literatura e aos resultados obtidos com o programa computacional de otimização Gurobi e também são analisados os tempos de processamento.

4.2.1 Análise dos resultados pelos limitantes inferiores

Para verificar o efeito da heurística diferença-de-quadrados DS com o aumento na quantidade de itens por instância foram utilizadas as instâncias de Delorme et al (2016) que formam um conjunto grande de instâncias, com quantidades de itens

variadas entre 50 e 1000 itens por instância, capacidades de contentores entre 50 e 1000, e pesos mínimos e máximos de itens variando entre $(0,1C; 0,2C)$ e $(0,7C; 0,8C)$ respectivamente. Para cada quadruplete há 10 instâncias.

Tabela 2 – Percentual de soluções ótimas e tempos de processamento (horas) versus número de itens na instância

	% de soluções ótimas								Tempo de processamento (horas)							
Heurística # Itens	50	100	200	300	400	500	750	1000	50	100	200	300	400	500	750	1000
BFD	44,8%	24,8%	14,0%	8,5%	6,3%	7,1%	5,4%	4,2%	0,0008	0,0017	0,0057	0,0098	0,0170	0,0265	0,0592	0,1116
BFD+DS	74,6%	61,7%	52,1%	44,8%	41,7%	39,6%	39,6%	39,8%	0,0023	0,0071	0,0282	0,0539	0,0927	0,1456	0,3151	0,5779
FFD	66,7%	43,8%	25,8%	18,3%	12,3%	13,8%	10,2%	9,0%	0,0006	0,0011	0,0038	0,0067	0,0117	0,0178	0,0395	0,0734
FFD+DS	74,8%	60,8%	52,5%	46,0%	42,3%	39,6%	38,8%	40,4%	0,0021	0,0059	0,0251	0,0490	0,0872	0,1348	0,2976	0,5482
MBS	21,5%	15,8%	15,0%	18,8%	18,1%	19,6%	19,6%	23,2%	0,0025	0,0085	0,0481	0,1086	0,2339	0,4279	1,3408	3,6646
MBS'	22,9%	15,6%	15,0%	18,8%	18,1%	19,6%	19,6%	23,2%	0,0024	0,0083	0,0471	0,1072	0,2305	0,4261	1,3421	3,6602
MBS+DS	21,5%	15,8%	15,0%	18,8%	18,1%	19,6%	19,6%	23,2%	0,0041	0,0129	0,0637	0,1346	0,2740	0,4965	1,4782	3,9076
MBSR	29,8%	19,2%	22,3%	24,2%	24,0%	24,4%	22,1%	26,2%	0,0018	0,0058	0,0286	0,0619	0,1359	0,2487	0,8791	2,7101
NFD	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0007	0,0016	0,0047	0,0078	0,0131	0,0207	0,0451	0,0850
NFD+DS	63,5%	46,5%	39,2%	32,5%	28,1%	28,8%	24,4%	28,9%	0,0041	0,0138	0,0574	0,1071	0,1876	0,2969	0,6618	1,2556
WS+DS	75,0%	62,1%	53,8%	46,5%	42,1%	40,2%	38,8%	41,6%	0,0045	0,0155	0,0725	0,1437	0,2712	0,4484	1,1629	2,3535
Geral	45,0%	33,3%	27,7%	25,2%	22,8%	22,9%	21,6%	23,6%	0,0260	0,0821	0,3848	0,7902	1,5544	2,6898	7,6213	18,9477

Tabela 3 – Diferença média e máxima entre o limitante superior e o inferior versus número de itens na instância

	Diferença média de contentores (LS-LI)								Diferença máxima de contentores (LS-LI)							
Heurística # Itens	50	100	200	300	400	500	750	1000	50	100	200	300	400	500	750	1000
BFD	0,56	0,83	1,18	1,51	1,91	2,10	2,76	3,49	2	2	4	5	6	7	8	10
BFD+DS	0,26	0,40	0,59	0,72	0,91	0,91	1,06	1,29	2	2	3	4	5	5	5	6
FFD	0,34	0,60	0,98	1,35	1,75	1,92	2,60	3,33	2	2	4	5	6	7	8	10
FFD+DS	0,25	0,41	0,58	0,70	0,91	0,91	1,06	1,28	2	2	3	4	5	5	5	6
MBS	1,36	2,44	3,76	4,39	5,30	5,38	6,23	6,95	6	6	15	19	27	27	26	37
MBS'	1,31	2,41	3,72	4,36	5,29	5,36	6,23	6,95	6	6	15	19	27	27	26	37
MBS+DS	1,36	2,44	3,76	4,39	5,30	5,38	6,23	6,95	6	6	15	19	27	27	26	37
MNSR	1,03	1,83	2,54	2,80	3,25	3,17	3,70	4,09	5	7	9	12	14	14	17	20
NFD	7,95	17,16	35,38	53,17	71,99	89,81	135,51	180,75	11	22	43	65	88	104	156	208
NFD+DS	0,38	0,62	0,90	1,20	1,61	1,61	2,06	2,45	2	3	4	7	7	8	12	15
WS+DS	0,25	0,39	0,56	0,69	0,88	0,89	1,02	1,21	2	2	3	3	5	5	5	7
Geral	1,37	2,68	4,91	6,84	9,01	10,68	15,32	19,89	11	22	43	65	88	104	156	208

Os resultados dos testes iniciais obtidos com as instâncias de Delorme et al (2016) estão listados na Tabela 2 e na Tabela 3.

Tabela 4 – Resultados dos experimentos computacionais

Heurística	Ótimos LS = LI	Valor médio de G	Tempo médio (seg)	Tempo total (horas)	% ótimos LS = LI	Diferença média entre LS e LI	Diferença máxima entre LS e LI
MBS+SA+DS	991	0,93123	0,5942	0,1997	81,90%	0,2190	6
MBSR+SA+DS	991	0,93120	0,3719	0,1250	81,90%	0,2182	4
MBS'+SA+DS	981	0,93108	0,5867	0,1972	81,07%	0,2248	3
FFD+SA+DS	979	0,92990	0,1705	0,0573	80,91%	0,2413	5
MBSR+DS+SA+DS	972	0,93057	0,0548	0,0184	80,33%	0,2504	5
FFD+DS+SA+DS	971	0,92941	0,0524	0,0176	80,25%	0,2744	6
MBS'+DS+SA+DS	971	0,93029	0,0545	0,0183	80,25%	0,2620	7
BFD+SA+DS	963	0,92964	0,1865	0,0627	79,59%	0,2545	5
FFD+DS+SA	960	0,92932	0,0522	0,0175	79,34%	0,2893	7
FFD+DS	959	0,92931	0,0150	0,0050	79,26%	0,2901	7
MBS+DS+SA+DS	959	0,93034	0,0583	0,0196	79,26%	0,2694	6
WS+DS+SA+DS	958	0,92949	0,0629	0,0211	79,17%	0,2810	7
SSD+SA+DS	953	0,92925	0,2356	0,0792	78,76%	0,2694	5
BFD+DS+SA+DS	951	0,92923	0,0538	0,0181	78,60%	0,2942	7
SSD+DS+SA+DS	949	0,92897	0,0533	0,0179	78,43%	0,2942	7
WS+DS+SA	948	0,92935	0,0567	0,0191	78,35%	0,2975	7
BFD+DS+SA	947	0,92913	0,0526	0,0177	78,26%	0,2992	7
WS+DS	946	0,92933	0,0219	0,0074	78,18%	0,3000	7
BFD+DS	944	0,92913	0,0162	0,0054	78,02%	0,3000	7
SSD+DS+SA	944	0,92893	0,0501	0,0168	78,02%	0,2983	7
SSD+DS	939	0,92888	0,0179	0,0060	77,60%	0,3017	7
MBS'+DS+SA	936	0,92986	0,0567	0,0191	77,36%	0,3140	8
MBSR+DS+SA	935	0,93009	0,0538	0,0181	77,27%	0,3182	9
MBSR+DS	932	0,92986	0,0182	0,0061	77,02%	0,3248	9
MBS'+DS	932	0,92954	0,0180	0,0060	77,02%	0,3314	7
MBS+DS+SA	929	0,92991	0,0542	0,0182	76,78%	0,3198	6
LFD+DS+SA+DS	929	0,92583	0,0626	0,0210	76,78%	0,3107	6
MBS+DS	928	0,92960	0,0184	0,0062	76,69%	0,3322	6
LFD+SA+DS	912	0,92603	0,2952	0,0992	75,37%	0,3132	6
WFD+SA+DS	901	0,92656	0,5917	0,1989	74,46%	0,3281	5
WFD+DS+SA+DS	897	0,92636	0,0680	0,0229	74,13%	0,3430	7
LFD+DS+SA	872	0,92523	0,0631	0,0212	72,07%	0,3893	6
WS+SA+DS	871	0,92963	6,1088	2,0532	71,98%	0,4339	8
LFD+DS	869	0,92481	0,0159	0,0053	71,82%	0,3950	6
NFD+SA+DS	867	0,92742	1,1529	0,3875	71,65%	0,4686	9
WFD+DS+SA	839	0,92582	0,0633	0,0213	69,34%	0,4149	7
WFD+DS	819	0,92541	0,0188	0,0063	67,69%	0,4463	7
FFD+SA	750	0,92463	0,1616	0,0543	61,98%	0,8628	20
FFD	736	0,92045	0,0108	0,0036	60,83%	0,9719	21

Tabela 5 – Resultados dos experimentos computacionais (continuação)

Heurística	Ótimos LS = LI	Valor médio de G	Tempo médio (seg)	Tempo total (horas)	% ótimos LS = LI	Diferença média entre LS e LI	Diferença máxima entre LS e LI
NFD+DS+SA+DS	724	0,92521	0,1399	0,0470	59,83%	0,8579	15
LFD+SA	636	0,91957	0,2745	0,0923	52,56%	1,0322	16
LFD	620	0,91381	0,0111	0,0037	51,24%	1,1496	21
BFD+SA	617	0,92367	0,1850	0,0622	50,99%	0,9711	16
NFD+DS+SA	590	0,92214	0,1268	0,0426	48,76%	1,3066	15
BFD	542	0,91895	0,0111	0,0037	44,79%	1,1463	22
NFD+DS	490	0,91587	0,0201	0,0067	40,50%	2,2331	21
MBSR+SA	437	0,92049	0,3787	0,1273	36,12%	3,8380	47
SSD+SA	436	0,91891	0,2314	0,0778	36,03%	2,1512	22
WFD+SA	419	0,91677	0,5727	0,1925	34,63%	1,5157	20
MBS+SA	387	0,91937	0,5699	0,1916	31,98%	4,4636	41
MBS'+SA	380	0,91799	0,5452	0,1832	31,40%	4,6281	44
WFD	338	0,90845	0,0107	0,0036	27,93%	1,7711	22
SSD	287	0,91139	0,0111	0,0037	23,72%	2,8868	24
MBSR	267	0,88873	0,0120	0,0040	22,07%	7,0281	94
NFD+SA	236	0,90698	1,0196	0,3427	19,50%	4,4736	58
MBS'	205	0,86698	0,0119	0,0040	16,94%	9,9512	88
MBS	200	0,86502	0,0121	0,0041	16,53%	10,0975	88
WS+SA	196	0,91362	5,7197	1,9224	16,20%	5,4050	39
NFD	59	0,78072	0,0108	0,0036	4,88%	21,7248	99
WS	0	0,41169	0,0020	0,0007	0,00%	130,5165	446

Pode ser verificado que a heurística DS melhora os resultados das soluções iniciais, diminuindo as diferenças média e máxima entre o limitante superior LS e o limitante inferior LI das instâncias testadas. Verificou-se também, como era esperado, que os tempos de processamento aumentam com o número de itens da instância. Para os testes finais com o *simulated annealing* foi processado um conjunto de instâncias propostas por Scholl et al (1997) composto por 1210 instâncias de níveis de dificuldade variados. Estas instâncias foram baixadas da página web de resultados dos autores (<https://www2.wiwi.uni-jena.de/Entscheidung/binpp/index.htm>).

As instâncias são divididas em três conjuntos com 720, 480 e 10 instâncias (conjuntos Scholl1, Scholl2 e Scholl3 respectivamente), com quantidades de itens entre 50 e 500. As capacidades dos contentores variam entre 100 e 150 (para o

conjunto Scholl1), são iguais a 100 (para o conjunto Scholl2), e são iguais a 100.000 (para o conjunto Scholl3).

Foram processadas 60 heurísticas geradas pelas combinações entre heurísticas geradoras de soluções iniciais (FFD, LFD, BFD, WFD, NFD, SSD, MBS, MBS', MBSR e WS) com e sem a heurística de busca local diferença-de-quadrados – DS, proposta como contribuição desta pesquisa e a metaheurística SA, executando as combinações SA, DS, SA+DS, DS+SA e DS+SA+DS com cada uma das soluções iniciais.

Os algoritmos assim gerados são designados pelas siglas HEUR, HEUR+DS, HEUR+SA, HEUR+DS+SA, HEUR+SA+DS e HEUR+DS+SA+DS, onde HEUR é uma das heurísticas (FFD, LFD, BFD, WFD, NFD, SSD, MBS, MBS', MBSR e WS).

Na Tabela 4 e na Tabela 5 estão apresentados os resultados obtidos. São listados o número de instâncias com ótimos comprovados (melhor limitante inferior LI igual ao limitante superior da heurística LS), o tempo médio de processamento por instância (em segundos), tempo total de processamento do conjunto de instâncias (em horas), percentual de soluções ótimas obtidas no conjunto das 1.210 instâncias processadas, diferença média e diferença máxima entre o melhor limitante inferior LI e o limitante superior LS da solução da heurística.

Tabela 6 – Resultados das heurísticas com solução inicial MBS

Heurística	Ótimos LS = LI	Valor médio de G	Tempo médio (seg)	Tempo total (horas)	% ótimos LS = LI	Diferença média entre LS e LI	Diferença máxima entre LS e LI
MBS+SA+DS	991	0,93123	0,5942	0,1997	81,90%	0,2190	6
MBS+DS+SA+DS	959	0,93034	0,0583	0,0196	79,26%	0,2694	6
MBS+DS+SA	929	0,92991	0,0542	0,0182	76,78%	0,3198	6
MBS+DS	928	0,92960	0,0184	0,0062	76,69%	0,3322	6
MBS+SA	387	0,91937	0,5699	0,1916	31,98%	4,4636	41
MBS	200	0,86502	0,0121	0,0041	16,53%	10,0975	88

Tabela 7 – Resultados das heurísticas com solução inicial MBS'

Heurística	Ótimos LS = LI	Valor médio de G	Tempo médio (seg)	Tempo total (horas)	% ótimos LS = LI	Diferença média entre LS e LI	Diferença máxima entre LS e LI
MBS'+SA+DS	981	0,93108	0,5867	0,1972	81,07%	0,2248	3
MBS'+DS+SA+DS	971	0,93029	0,0545	0,0183	80,25%	0,2620	7
MBS'+DS+SA	936	0,92986	0,0567	0,0191	77,36%	0,3140	8
MBS'+DS	932	0,92954	0,0180	0,0060	77,02%	0,3314	7
MBS'+SA	380	0,91799	0,5452	0,1832	31,40%	4,6281	44
MBS'	205	0,86698	0,0119	0,0040	16,94%	9,9512	88

Os resultados indicam que as heurísticas propostas por Fleszar e Hindi (2002) combinadas com o *simulated annealing* para melhoria da solução e a busca local com a heurística diferença-de-quadrados DS aplicada ao resultado do *simulated annealing* apresentam o maior número de soluções ótimas. Na heurística MBS+SA+DS, a melhor do conjunto das 60 processadas, 81,90% de soluções ótimas foram obtidas.

Na Tabela 6 são apresentados os resultados para as diferentes combinações geradas pelas heurísticas com solução inicial MBS e com número máximo de iterações sem melhorias igual a 50 e a primeira agenda de parâmetros do *simulated annealing* com $\sigma = 0,97$. Estes foram os melhores resultados obtidos com estes parâmetros. Verificou-se que a heurística MBS+SA+DS obteve 991 soluções ótimas (limitante superior LS igual ao melhor limitante inferior LI) das 1210 instâncias de Scholl et al (1997), o que representa 81,90% de soluções ótimas. Levando em média 0,5942 segundos de processamento por instância e tempo total de 0,1997 horas para processamento de todas as instâncias. Pela Tabela 6 pode ser observada a influência que cada combinação de heurísticas tem no conjunto do algoritmo. A solução inicial MBS obtêm apenas 200 soluções ótimas (16,53% do total).

Quando a metaheurística SA é aplicada após a MBS o número de soluções ótimas sobe para 387 (31,98% do total). Ao aplicar a heurística DS após o MBS o número de soluções ótimas subiu para 928 (76,69% do total). Ao aplicar DS+SA após o MBS o número de soluções ótimas totaliza 929 (76,78%), sendo este o terceiro melhor resultado com a solução inicial da heurística MBS.

Ao aplicar a heurística proposta nesta pesquisa ao resultado do MBS+DS+SA, gerando a heurística MBS+DS+SA+DS o número de soluções ótimas passa de 929 para 959 (79,26% do total de instâncias), este foi o segundo melhor resultado com a solução inicial MBS.

Na Tabela 7 são apresentados os resultados com solução inicial gerada pela heurística MBS' que obtém 981 soluções ótimas quando combinada com *Simulated Annealing* – SA e Diferença-de-Quadrados – DS formando a heurística MBS'+SA+DS, sendo este o melhor resultado obtido quando a solução inicial é MBS' e a agenda de parâmetros é a número 1 com número máximo de iterações sem melhoria igual a 50 e $\sigma = 0,97$.

Na Tabela 8 estão apresentados os resultados obtidos quando a solução inicial é gerada pela heurística MBSR proposta por Fleszar e Hindi (2002). O melhor resultado obtido com esta solução inicial leva 0,1250 horas gerando 991 soluções ótimas (limitante superior LS igual ao limitante inferior LI), o que representa 81,90% do total de instâncias. Os resultados se igualam aos do MBS, porém com um valor médio da função objetivo menor, o que indica que a ocupação dos contentores fica menor que a obtida pela MBS. No entanto ao compararmos os tempos de processamento verificamos que, conforme conclusão de Fleszar e Hindi (2002), MBSR tem menor tempo de processamento que MBS.

Tabela 8 – Resultados das heurísticas com solução inicial MBSR

Heurística	Ótimos LS = LI	Valor médio de G	Tempo médio (seg)	Tempo total (horas)	% ótimos LS = LI	Diferença média entre LS e LI	Diferença máxima entre LS e LI
MBSR+SA+DS	991	0,93120	0,3719	0,1250	81,90%	0,2182	4
MBSR+DS+SA+DS	972	0,93057	0,0548	0,0184	80,33%	0,2504	5
MBSR+DS+SA	935	0,93009	0,0538	0,0181	77,27%	0,3182	9
MBSR+DS	932	0,92986	0,0182	0,0061	77,02%	0,3248	9
MBSR+SA	437	0,92049	0,3787	0,1273	36,12%	3,8380	47
MBSR	267	0,88873	0,0120	0,0040	22,07%	7,0281	94

A Tabela 9 exhibe os resultados com a solução inicial pelo FFD. O melhor resultado é 979 soluções ótimas (80,91% do total).

Tabela 9 – Resultados das heurísticas com solução inicial FFD

Heurística	Ótimos LS = LI	Valor médio de G	Tempo médio (seg)	Tempo total (horas)	% ótimos LS = LI	Diferença média entre LS e LI	Diferença máxima entre LS e LI
FFD+SA+DS	979	0,92990	0,1705	0,0573	80,91%	0,2413	5
FFD+DS+SA+DS	971	0,92941	0,0524	0,0176	80,25%	0,2744	6
FFD+DS+SA	960	0,92932	0,0522	0,0175	79,34%	0,2893	7
FFD+DS	959	0,92931	0,0150	0,0050	79,26%	0,2901	7
FFD+SA	750	0,92463	0,1616	0,0543	61,98%	0,8628	20
FFD	736	0,92045	0,0108	0,0036	60,83%	0,9719	21

Os resultados gerados com a solução inicial pelo BFD são apresentados na Tabela 10, novamente a combinação HEUR+SA+DS gera os melhores resultados.

Tabela 10 – Resultados das heurísticas com solução inicial BFD

Heurística	Ótimos LS = LI	Valor médio de G	Tempo médio (seg)	Tempo total (horas)	% ótimos LS = LI	Diferença média entre LS e LI	Diferença máxima entre LS e LI
BFD+SA+DS	963	0,92964	0,1865	0,0627	79,59%	0,2545	5
BFD+DS+SA+DS	951	0,92923	0,0538	0,0181	78,60%	0,2942	7
BFD+DS+SA	947	0,92913	0,0526	0,0177	78,26%	0,2992	7
BFD+DS	944	0,92913	0,0162	0,0054	78,02%	0,3000	7
BFD+SA	617	0,92367	0,1850	0,0622	50,99%	0,9711	16
BFD	542	0,91895	0,0111	0,0037	44,79%	1,1463	22

Tabela 11 – Resultados das heurísticas com solução inicial WS

Heurística	Ótimos LS = LI	Valor médio de G	Tempo médio (seg)	Tempo total (horas)	% ótimos LS = LI	Diferença média entre LS e LI	Diferença máxima entre LS e LI
WS+DS+SA+DS	958	0,92949	0,0629	0,0211	79,17%	0,2810	7
WS+DS+SA	948	0,92935	0,0567	0,0191	78,35%	0,2975	7
WS+DS	946	0,92933	0,0219	0,0074	78,18%	0,3000	7
WS+SA+DS	871	0,92963	6,1088	2,0532	71,98%	0,4339	8
WS+SA	196	0,91362	5,7197	1,9224	16,20%	5,4050	39
WS	0	0,41169	0,0020	0,0007	0,00%	130,5165	446

Na Tabela 11 podem ser comparados os resultados gerados com solução inicial pelo WS. Neste caso o melhor resultado vem da combinação WS+DS+SA+DS. Verifica-se que WS+DS+SA+DS gera 10 soluções ótimas a mais que a WS+DS+SA.

Comparando os resultados do WS+DS+SA com os do WS+SA+DS podemos verificar que a heurística DS reduz o tempo de processamento quando aplicada antes do SA, levando 0,93% do tempo de processamento e obtendo 8,48% mais soluções ótimas.

Na Tabela 12 estão listados os resultados gerados com a solução inicial pela SSD que obtêm 953 soluções ótimas em 0,0792 horas de processamento. Para esta solução inicial os melhores resultados são obtidos com a combinação de heurísticas SSD+SA+DS. Os resultados indicam que a aplicação da heurística DS antes da SA reduzem o tempo de processamento, mas não aumentam o número de soluções ótimas obtidas, obtendo 9 soluções ótimas a menos, porém com 21% do tempo de processamento.

Tabela 12 – Resultados das heurísticas com solução inicial SSD

Heurística	Ótimos LS = LI	Valor médio de G	Tempo médio (seg)	Tempo total (horas)	% ótimos LS = LI	Diferença média entre LS e LI	Diferença máxima entre LS e LI
SSD+SA+DS	953	0,92925	0,2356	0,0792	78,76%	0,2694	5
SSD+DS+SA+DS	949	0,92897	0,0533	0,0179	78,43%	0,2942	7
SSD+DS+SA	944	0,92893	0,0501	0,0168	78,02%	0,2983	7
SSD+DS	939	0,92888	0,0179	0,0060	77,60%	0,3017	7
SSD+SA	436	0,91891	0,2314	0,0778	36,03%	2,1512	22
SSD	287	0,91139	0,0111	0,0037	23,72%	2,8868	24

Na Tabela 13 são apresentados os resultados com solução pelo LFD. Diferente da maioria das outras soluções iniciais, para o LFD a melhor combinação é da forma HEUR+DS+SA+DS (929 soluções ótimas), seguido do LFD+SA+DS com 912 soluções ótimas.

Tabela 13 – Resultados das heurísticas com solução inicial LFD

Heurística	Ótimos LS = LI	Valor médio de G	Tempo médio (seg)	Tempo total (horas)	% ótimos LS = LI	Diferença média entre LS e LI	Diferença máxima entre LS e LI
LFD+DS+SA+DS	929	0,92583	0,0626	0,0210	76,78%	0,3107	6
LFD+SA+DS	912	0,92603	0,2952	0,0992	75,37%	0,3132	6
LFD+DS+SA	872	0,92523	0,0631	0,0212	72,07%	0,3893	6
LFD+DS	869	0,92481	0,0159	0,0053	71,82%	0,3950	6
LFD+SA	636	0,91957	0,2745	0,0923	52,56%	1,0322	16
LFD	620	0,91381	0,0111	0,0037	51,24%	1,1496	21

Tabela 14 – Resultados das heurísticas com solução inicial WFD

Heurística	Ótimos LS = LI	Valor médio de G	Tempo médio (seg)	Tempo total (horas)	% ótimos LS = LI	Diferença média entre LS e LI	Diferença máxima entre LS e LI
WFD+SA+DS	901	0,92656	0,5917	0,1989	74,46%	0,3281	5
WFD+DS+SA+DS	897	0,92636	0,0680	0,0229	74,13%	0,3430	7
WFD+DS+SA	839	0,92582	0,0633	0,0213	69,34%	0,4149	7
WFD+DS	819	0,92541	0,0188	0,0063	67,69%	0,4463	7
WFD+SA	419	0,91677	0,5727	0,1925	34,63%	1,5157	20
WFD	338	0,90845	0,0107	0,0036	27,93%	1,7711	22

Os resultados da Tabela 14 para o WFD mostram que 901 soluções ótimas são obtidas em 0,1989 horas de processamento com a combinação WFD+SA+DS.

Tabela 15 – Resultados das heurísticas com solução inicial NFD

Heurística	Ótimos LS = LI	Valor médio de G	Tempo médio (seg)	Tempo total (horas)	% ótimos LS = LI	Diferença média entre LS e LI	Diferença máxima entre LS e LI
NFD+SA+DS	867	0,92742	1,1529	0,3875	71,65%	0,4686	9
NFD+DS+SA+DS	724	0,92521	0,1399	0,0470	59,83%	0,8579	15
NFD+DS+SA	590	0,92214	0,1268	0,0426	48,76%	1,3066	15
NFD+DS	490	0,91587	0,0201	0,0067	40,50%	2,2331	21
NFD+SA	236	0,90698	1,0196	0,3427	19,50%	4,4736	58
NFD	59	0,78072	0,0108	0,0036	4,88%	21,7248	99

Quando a solução inicial é gerada pela heurística NFD, a melhor combinação é gerada pela heurística NFD+SA+DS com 867 soluções ótimas processadas em 0,3875 horas.

4.2.2 Análise do tempo de processamento

Nesta seção são descritas as análises da influência dos parâmetros nos tempos de processamento.

Para verificar a influência de cada parte do algoritmo (cálculo dos limitantes inferiores LI, processamento da solução inicial HEUR, processamento da heurística diferença-de-quadrados DS e do *simulated annealing* SA) no tempo total de processamento, foram medidos e comparados os tempos totais de processamento das 1210 instâncias utilizadas com números máximos de iterações sem melhorias do *simulated annealing* iguais a 50, 100 e 200.

O efeito de cada parte do algoritmo no tempo total de processamento pode ser visualizado no gráfico da Figura 29 quando o número máximo de iterações sem melhorias é definido como 50 e a agenda de parâmetros é a número 1.

Tabela 16 – Tempo total de processamento (segundos) por parte do algoritmo com número máximo de iterações sem melhorias igual a 50

Heurística	LI	INICIAL	DS	SA	TOTAL
WS	320,01	44,56	67,35	80,83	512,75
NFD	320,01	15,00	13,11	279,21	627,32
FFD	320,01	15,90	5,76	94,39	436,05
LFD	320,01	10,59	47,48	50,54	428,62
BFD	320,01	15,41	52,99	19,76	408,17
WFD	320,01	10,66	5,80	117,57	454,03
SSD	320,01	22,07	16,89	68,73	427,69
MBS	320,01	17,45	17,46	88,02	442,94
MBS'	320,01	14,09	15,92	97,85	447,85
MBSR	320,01	17,85	17,59	58,53	413,97

Figura 29 – Tempo total de processamento (segundos) por heurística inicial com número máximo de iterações sem melhoria igual a 50

Os dados estão listados na Tabela 16, onde para cada heurística foram registrados os tempos de processamento total das 1210 instâncias de Scholl et al (1997) em segundos com número máximo de iterações sem melhoria no *simulated annealing* igual a 50 e a primeira agenda de parâmetros.

O gráfico da Figura 30 preenchido com os dados da Tabela 17, mostra que para todos os tipos de soluções iniciais a maior parte do tempo de processamento (em média 69,6% do tempo total) está concentrado na parte LI, que representa os tempos de processamento do cálculo dos limitantes inferiores. A segunda parte com maior tempo de processamento é o *simulated annealing* (em média com 20,8% do tempo total). A terceira maior parte do algoritmo é dedicada ao processamento da

heurística proposta nesta pesquisa, a heurística diferença-de-quadrados (com 5,7% do tempo total). Por último fica a parte do algoritmo dedicada a geração da solução inicial (com 4,0% do tempo total).

Figura 30 – Percentual do tempo total de processamento por tipo de solução inicial e parte do algoritmo com número máximo de iterações sem melhorias igual a 50

Pela Tabela 17 podemos verificar que analisando as heurísticas individualmente as proporções no tempo de processamento variam, mas em geral mantêm a mesma ordem das partes do algoritmo. Isso ocorre porque o cálculo dos limitantes inferiores L_3 é custoso em termos de tempo de processamento. O *simulated annealing* também tem alto tempo de processamento se comparado às demais partes do algoritmo. Por outro lado, as heurísticas geradoras de solução inicial e de melhoria dela, a heurística diferença-de-quadrados são menos custosas computacionalmente.

Quando aumentamos o limite máximo de iterações sem melhorias no *simulated annealing* de 50 para 100, aumentamos consideravelmente o tempo de

processamento desta parte do algoritmo. Com essa alteração o percentual médio no tempo total de processamento do *simulated annealing* sobe de 20,8% (com número máximo de iterações sem melhoria igual a 50) para 49,2%.

Tabela 17 – Percentual do tempo de processamento por tipo de solução inicial e parte do algoritmo com número máximo de iterações igual a 50

HEUR	LI	INICIAL	DS	SA	TOTAL
WS	62,4%	8,7%	13,1%	15,8%	100,0%
NFD	51,0%	2,4%	2,1%	44,5%	100,0%
FFD	73,4%	3,6%	1,3%	21,6%	100,0%
LFD	74,7%	2,5%	11,1%	11,8%	100,0%
BFD	78,4%	3,8%	13,0%	4,8%	100,0%
WFD	70,5%	2,3%	1,3%	25,9%	100,0%
SSD	74,8%	5,2%	3,9%	16,1%	100,0%
MBS	72,2%	3,9%	3,9%	19,9%	100,0%
MBS'	71,5%	3,1%	3,6%	21,8%	100,0%
MBSR	77,3%	4,3%	4,2%	14,1%	100,0%

O impacto da alteração pode ser observado na Figura 31 e na Figura 32, onde agora o *simulated annealing* passa a representar quase a metade do tempo de processamento, ou até mais da metade quando a solução inicial for pela heurística NFD.

Tabela 18 – Tempo total de processamento (segundos) por parte do algoritmo com número máximo de iterações sem melhorias igual a 100

Heurística	LI	HEUR	DS	SA	TOTAL
WS	320,01	44,56	67,35	124,77	556,68
NFD	320,01	15,00	13,11	1162,06	1510,18
FFD	320,01	15,90	5,76	211,75	553,42
LFD	320,01	10,59	47,48	268,28	646,35
BFD	320,01	15,41	52,99	120,02	508,43
WFD	320,01	10,66	5,80	414,81	751,27
SSD	320,01	22,07	16,89	165,33	524,29
MBS	320,01	17,45	17,46	336,45	691,37
MBS'	320,01	14,09	15,92	439,27	789,27
MBSR	320,01	17,85	17,59	288,31	643,75

Figura 31 – Tempo total de processamento (em segundos) por parte do algoritmo com número máximo de iterações sem melhoria igual a 100

Figura 32 – Percentual do tempo total de processamento por parte do algoritmo com número máximo de iterações sem melhorias igual a 100

Quando o número máximo de iterações sem melhoria é parametrizado como 200 o tempo de processamento do *simulated annealing* aumenta ainda mais e esta parte do algoritmo passa a representar em média 80,1% do tempo total de processamento conforme apresentado na Tabela 20 e Tabela 21.

Tabela 19 – Percentual do tempo de processamento por tipo de solução inicial e parte do algoritmo com número máximo de iterações igual a 100

Heurística	LI	HEUR	DS	SA	TOTAL
WS	57,5%	8,0%	12,1%	22,4%	100,0%
NFD	21,2%	1,0%	0,9%	76,9%	100,0%
FFD	57,8%	2,9%	1,0%	38,3%	100,0%
LFD	49,5%	1,6%	7,3%	41,5%	100,0%
BFD	62,9%	3,0%	10,4%	23,6%	100,0%
WFD	42,6%	1,4%	0,8%	55,2%	100,0%
SSD	61,0%	4,2%	3,2%	31,5%	100,0%
MBS	46,3%	2,5%	2,5%	48,7%	100,0%
MBS'	40,5%	1,8%	2,0%	55,7%	100,0%
MBSR	49,7%	2,8%	2,7%	44,8%	100,0%

A Figura 33 e a Figura 34 apresentam gráficos destes resultados onde é possível comparar a influência de cada parte do algoritmo para cada tipo de solução inicial.

Tabela 20 – Tempo total de processamento (segundos) por parte do algoritmo com número máximo de iterações sem melhorias igual a 200

Heurística	LI	HEUR	DS	SA	TOTAL
WS	320,01	44,56	67,35	879,66	1311,58
NFD	320,01	15,00	13,11	4043,45	4391,57
FFD	320,01	15,90	5,76	899,16	1240,83
LFD	320,01	10,59	47,48	1084,92	1463,00
BFD	320,01	15,41	52,99	978,37	1366,78
WFD	320,01	10,66	5,80	973,58	1310,04
SSD	320,01	22,07	16,89	674,68	1033,64
MBS	320,01	17,45	17,46	1706,69	2061,60
MBS'	320,01	14,09	15,92	1641,15	1991,16
MBSR	320,01	17,85	17,59	1769,10	2124,54

Figura 33 – Tempo total de processamento (em segundos) por parte do algoritmo com número máximo de iterações sem melhoria igual a 200

Com os resultados destas análises o algoritmo do simulated annealing foi parametrizado com número máximo de iterações sem melhorias como 50, pois este apresenta a melhor alternativa testada em termos da relação de tempos de processamento e número de soluções ótimas geradas.

Tabela 21 – Percentual do tempo de processamento por tipo de solução inicial e parte do algoritmo com número máximo de iterações igual a 200

Heurística	LI	HEUR	DS	SA	TOTAL
WS	24,4%	3,4%	5,1%	67,1%	100,0%
NFD	7,3%	0,3%	0,3%	92,1%	100,0%
FFD	25,8%	1,3%	0,5%	72,5%	100,0%
LFD	21,9%	0,7%	3,2%	74,2%	100,0%
BFD	23,4%	1,1%	3,9%	71,6%	100,0%
WFD	24,4%	0,8%	0,4%	74,3%	100,0%
SSD	31,0%	2,1%	1,6%	65,3%	100,0%
MBS	15,5%	0,8%	0,8%	82,8%	100,0%
MBS'	16,1%	0,7%	0,8%	82,4%	100,0%
MBSR	15,1%	0,8%	0,8%	83,3%	100,0%

Figura 34 – Percentual do tempo total de processamento por parte do algoritmo com número máximo de iterações sem melhorias igual a 200

4.2.3 Teste de robustez do *simulated annealing*

Para testar a robustez das heurísticas com *simulated annealing* propostas na pesquisa os algoritmos foram processados 10 vezes e o tempo médio de processamento e o valor médio da função objetivo revisada foram tabulados.

Foi calculado o desvio padrão dos tempos de processamento por heurística e quantidade de itens da instância processada. Para esta análise foram utilizadas as heurísticas do tipo HEUR+DS+SA onde HEUR é uma das 10 heurísticas da literatura utilizadas para gerar as soluções iniciais (FFD, LFD, WFD, BFD, NFD, SSD, MBS, MBS', MBSR e WS).

Na Tabela 22 são listados o valor médio e desvio padrão da função objetivo revisada, quantidades de instâncias com limitante superior LS igual ao limitante inferior LI (# ótimos), total de instâncias utilizadas e tempo médio e desvio padrão do tempo de processamento por heurística e quantidade de itens da instância.

Tabela 22 – Resultados do teste de robustez das heurísticas do tipo HEUR+DS+SA

Itens	Heurística	G médio	desvio G	# ótimos	Instâncias	tempo médio (seg)	Desvio tempo (seg)	% ótimos
50	WS+DS+SA	0,8878	$2,3648 \times 10^{-14}$	269	300	0,00353	0,00458	89,67%
	FFD+DS+SA	0,8880	$2,3537 \times 10^{-14}$	272	300	0,01308	0,00470	90,67%
	NFD+DS+SA	0,8786	$2,2982 \times 10^{-14}$	216	300	0,01743	0,00978	72,00%
	LFD+DS+SA	0,8808	$2,3315 \times 10^{-14}$	260	300	0,01318	0,00418	86,67%
	BFD+DS+SA	0,8876	$2,2871 \times 10^{-14}$	268	300	0,01305	0,00432	89,33%
	WFD+DS+SA	0,8802	$2,6534 \times 10^{-14}$	250	300	0,01373	0,00505	83,33%
	SSD+DS+SA	0,8866	$2,6534 \times 10^{-14}$	264	300	0,01335	0,00472	88,00%
	MBS+DS+SA	0,8888	$2,2982 \times 10^{-14}$	258	300	0,01383	0,00471	86,00%
	MBS'+DS+SA	0,8883	$2,4425 \times 10^{-14}$	256	300	0,01366	0,00507	85,33%
	MBSR+DS+SA	0,8890	$2,2315 \times 10^{-14}$	254	300	0,01392	0,00542	84,67%
100	WS+DS+SA	0,9244	$2,6645 \times 10^{-14}$	252	300	0,01081	0,02066	84,00%
	FFD+DS+SA	0,9243	$2,3537 \times 10^{-14}$	253	300	0,01915	0,02239	84,33%
	NFD+DS+SA	0,9174	$2,5091 \times 10^{-14}$	170	300	0,03682	0,03298	56,67%
	LFD+DS+SA	0,9195	$2,7534 \times 10^{-14}$	232	300	0,02108	0,02058	77,33%
	BFD+DS+SA	0,9240	$2,2982 \times 10^{-14}$	248	300	0,02033	0,02788	82,67%
	WFD+DS+SA	0,9202	$2,6090 \times 10^{-14}$	225	300	0,02257	0,02648	75,00%
	SSD+DS+SA	0,9239	$2,5202 \times 10^{-14}$	248	300	0,02029	0,02375	82,67%
	MBS+DS+SA	0,9251	$2,1982 \times 10^{-14}$	240	300	0,02166	0,02442	80,00%
	MBS'+DS+SA	0,9250	$2,3315 \times 10^{-14}$	244	300	0,02018	0,02118	81,33%
	MBSR+DS+SA	0,9255	$2,4203 \times 10^{-14}$	244	300	0,02141	0,03924	81,33%
200	WS+DS+SA	0,9453	$2,5313 \times 10^{-14}$	224	310	0,03457	0,05089	72,26%
	FFD+DS+SA	0,9453	$2,4869 \times 10^{-14}$	231	310	0,03719	0,04839	74,52%
	NFD+DS+SA	0,9398	$2,5979 \times 10^{-14}$	125	310	0,10722	0,11783	40,32%
	LFD+DS+SA	0,9423	$2,6534 \times 10^{-14}$	200	310	0,04808	0,05520	64,52%
	BFD+DS+SA	0,9452	$2,2315 \times 10^{-14}$	228	310	0,03905	0,04957	73,55%
	WFD+DS+SA	0,9435	$2,6756 \times 10^{-14}$	194	310	0,04915	0,05272	62,58%
	SSD+DS+SA	0,9453	$2,7089 \times 10^{-14}$	223	310	0,04130	0,04681	71,94%
	MBS+DS+SA	0,9456	$2,4758 \times 10^{-14}$	224	310	0,04434	0,06028	72,26%
	MBS'+DS+SA	0,9454	$2,5979 \times 10^{-14}$	225	310	0,04016	0,04922	72,58%
	MBSR+DS+SA	0,9457	$2,4314 \times 10^{-14}$	227	310	0,04348	0,06011	73,23%
500	WS+DS+SA	0,9592	$2,3981 \times 10^{-14}$	200	300	0,20370	0,29417	66,67%
	FFD+DS+SA	0,9592	$2,3870 \times 10^{-14}$	204	300	0,15829	0,24339	68,00%
	NFD+DS+SA	0,9540	$2,3759 \times 10^{-14}$	94	300	0,60647	0,75247	31,33%
	LFD+DS+SA	0,9575	$2,5979 \times 10^{-14}$	176	300	0,23752	0,34005	58,67%
	BFD+DS+SA	0,9591	$2,4536 \times 10^{-14}$	200	300	0,16727	0,24377	66,67%
	WFD+DS+SA	0,9585	$2,5202 \times 10^{-14}$	173	300	0,21555	0,26162	57,67%
	SSD+DS+SA	0,9595	$2,3537 \times 10^{-14}$	211	300	0,15434	0,23302	70,33%
	MBS+DS+SA	0,9598	$2,5313 \times 10^{-14}$	210	300	0,17937	0,27520	70,00%
	MBS'+DS+SA	0,9598	$2,5424 \times 10^{-14}$	212	300	0,18989	0,33579	70,67%
	MBSR+DS+SA	0,9600	$2,4425 \times 10^{-14}$	213	300	0,19317	0,33553	71,00%

Os resultados indicam que o valor médio da função objetivo não mudou significativamente de uma rodada para a outra. O número de contentores utilizados também foi o mesmo em todas as 10 rodadas de processamento.

O desvio padrão do tempo de processamento foi alto porque entre as instâncias com mesmo número de itens, há diferentes níveis de dificuldade. Como pode ser observado na Tabela 23, onde são tabelados os tempos de processamento para cada uma das 10 rodadas nas instâncias N4C2W2_P e N4W3B1R3, cada uma com 500 itens.

Tabela 23 – Tempo médio de processamento por rodada para duas instâncias de 500 itens

Tempo de processamento por rodada (segundos)													
Instancia	Heuristica	1	2	3	4	5	6	7	8	9	10	média	desvio
N4C2W2_P	NFD+DS+SA	5,429	5,475	5,351	5,507	5,116	5,366	6,755	5,242	7,254	6,537	5,8032	0,711864
N4W3B1R3	NFD+DS+SA	0,031	0,031	0,031	0,031	0,031	0,031	0,032	0,031	0,031	0,031	0,0311	0,0003

Os resultados da Tabela 23 indicam que para instâncias com a mesma quantidade de itens há diferentes níveis de dificuldade, o que aumenta a diferença entre os tempos de processamento e o desvio padrão dos tempos. Para a instância N4C2W2_P o tempo médio foi de 5,8032 segundos com desvio padrão de 0,711864 segundos, enquanto que para a instância N4W3B1R3 o tempo médio foi de 0,0311 segundos com desvio padrão de 0,0003 segundos. Isso indica que o desvio padrão nos tempos de processamento será alto quando calculado para instâncias diferentes com o mesmo número de itens.

5 CONCLUSÕES E FUTUROS DESENVOLVIMENTOS

O objetivo do estudo consistiu em obter o menor número possível de contentores necessários para alocar um dado número de itens. Sendo este um problema muito comum na indústria, tendo, por exemplo, aplicação no problema de carregamento de veículos em uma dimensão, onde a frota deve ser minimizada e um dado número de itens deve ser carregado respeitando a capacidade máxima do veículo.

Uma heurística simples de troca de itens entre contentores foi proposta para melhorar soluções iniciais geradas por heurísticas comuns para o problema de empacotamento unidimensional. Foi dado o nome de diferença-de-quadrados a heurística proposta devido às inequações necessárias para a melhoria da função objetivo revisada, inspirando a estrutura da heurística.

O procedimento consiste em um *loop* que testa todas as trocas de pares de itens dos tipos de inserção (inserir um item de um contentor em outro) e de inter-troca (troca entre de itens entre contentores) que geram melhorias na função objetivo. Todos os pares de itens são avaliados a cada iteração da busca e um contador de itens alterados na iteração anterior serve de critério de parada do algoritmo, sendo este finalizado na iteração em que o contador for nulo.

A cada iteração o número de contentores foi comparado a limitantes inferiores encontrados na literatura e, no caso em que foram iguais o algoritmo parou, pois uma solução ótima foi encontrada.

Quando a heurística diferença-de-quadrados é finalizada e o limitante inferior não é atingido, o algoritmo *simulated annealing* é acionado para melhorar a solução.

Desta forma foram geradas 60 combinações de heurísticas com e sem diferença-de-quadrados e o *simulated annealing* combinados com heurísticas consagradas de empacotamento unidimensional (FFD, LFD, BFD, WFD, NFD, SSD, MBS, MBS' e MBSR) e WS – a qual configura a pior solução inicial possível, com um item em cada contentor.

Os parâmetros do *simulated annealing* (comprimento de Markov e temperaturas inicial e final) foram calculados em função de dados do problema como número de itens da instância, capacidades dos contentores e limitantes inferiores, com o intuito de parametrizar o algoritmo de forma diferente para cada instância. Como os parâmetros utilizados na pesquisa dependiam somente de dados do problema, foi possível identificar uma constante que pode ser comparada a constante de Boltzman da termodinâmica. Sendo estas outras contribuições desta pesquisa.

As soluções vizinhas foram geradas através de trocas viáveis e aleatórias entre contentores dos tipos inserção, inter-troca, troca de um item por um par de itens e troca entre dois pares de itens. Movimentos inviáveis também são realizados e os itens sobressalentes do contentor com capacidade excedida são retirados do contentor e alocados novamente na solução com o algoritmo *First Fit*, tornando a solução viável novamente.

A diferença entre limitantes inferiores e superiores e os tempos de processamento foram usados para avaliação da qualidade das soluções geradas em experimentos computacionais realizados com 1210 instâncias de diferentes níveis de dificuldade propostas por Scholl et al (1997).

Os resultados indicaram que as heurísticas propostas por Fleszar e Hindi (2002) combinadas com o *simulated annealing* para melhoria da solução e a busca local com a heurística diferença-de-quadrados DS aplicada ao resultado do *simulated annealing* apresentaram o maior número de soluções ótimas.

As combinações de heurísticas MBS+SA+DS e MBSR+SA+DS, as melhores do conjunto das 60 processadas geraram 81,90% de soluções ótimas com tempos de processamento acessíveis à tomada de decisão em problemas de carregamentos de veículos. Ao combinar DS e SA com as heurísticas consagradas foram obtidas melhorias em todos os 10 tipos de solução inicial.

Outras heurísticas apresentadas na literatura como WABP e HI_BP obtiveram melhores resultados que as heurísticas propostas nessa pesquisa, principalmente em relação aos tempos de processamento. Por isso, como trabalhos futuros, pretende-se melhorar os tempos de processamento, os limitantes inferiores e as estruturas de vizinhança do *simulated annealing* para obter soluções ótimas em conjuntos de instâncias mais difíceis e desafiadores encontradas na literatura e melhorar os tempos de processamentos das heurísticas que foram apresentadas nesta pesquisa.

Também se pretende testar combinações da heurística Diferença-de-Quadrados – DS com outras metaheurísticas como VNS e ILS e verificar o que faz com que instâncias de mesmo tamanho sejam mais difíceis de resolver do que outras.

REFERÊNCIAS

- ABUHASSAN, I. A.; NASEREDDIN, H. H. O. Cutting stock problem: solution behaviors. **International Journal of Research and Reviews in Applied Sciences**, Jordan, Abril 2011. 429-433.
- AKTIN, T.; ÖZDEMİR, R. G. An integrated approach to the one-dimensional cutting stock problem in coronary stent manufacturing. **European Journal of Operational Research**, v. 196, n. 2, p. 737–743, Julho 2009.
- ALVIM, A. C.; RIBEIRO, C. C.; GLOVER, J. A hybrid improvement heuristic for the bin-packing problem and its application to the multiprocessor scheduling problem. **Journal of Heuristics**, v. 10, n. 2, p. 205–229, Março 2004.
- ASGEIRSSON, E. et al. Closed On-line Bin Packing. **Acta Cybernetica**, v. 15, n. 3, p. 361-367, Setembro 2002.
- ATIQULLAH, M. M. **An Efficient Simple Cooling Schedule for Simulated Annealing**. International Conference on Computational Science and Its Applications - ICCSA 2004. Assisi, Italy: [s.n.]. 2004. p. 396-404.
- BEASLEY, J. E. OR-Library: Distributing Test Problems by Electronic Mail. **The Journal of the Operational Research Society**, v. 41, n. 11, p. 1069-1072, Novembro 1990.
- BÉKÉSI, J.; GALAMBOS, G.; KELLERER, H. A $5/4$ linear time bin packing algorithm. **Journal of Computer and System Sciences**, v. 60, n. 1, p. 145-160, Fevereiro 2000.
- BENDER, M. A. et al. Sum-of-Squares Heuristics for Bin Packing and Memory Allocation. **Journal of the ACM (JACM)**, v. 2, n. 3, p. 1-19, Setembro 2001.
- BERBERLER, M. E.; NURIUEV, U.; YILDIRIM, A. A software for the one-dimensional cutting stock problem. **Journal of King Saud University - Science**, v. 23, n. 1, p. 69–76, Janeiro 2011.
- BRUSCO, M. J.; THOMPSON, G. M.; JACOBS, L. W. A Morph-Based Simulated Annealing Heuristic for a Modified Bin-Packing Problem. **Journal of the Operational Research Society**, v. 48, n. 4, p. 433-439, Abril 1997.
- BULJUBAŠIĆ, M.; VASQUEZ, M. Consistent neighborhood search for one-dimensional bin packing and two-dimensional vector packing. **Computers & Operations Research**, v. 76, p. 12-21, Dezembro 2016.
- BURKE, E. K.; HYDE, M. R.; KENDALL, G. Providing a Memory Mechanism to Enhance the Evolutionary Design of Heuristics. **IEEE Explore Digital Library**, p. 1 - 8, 2010. ISSN 10.1109/CEC.2010.5586388.
- CAPRARA, A. et al. Friendly bin packing instances without Integer Round-up Property. **Mathematical Programming**, v. 150, n. 1, p. 5-17, Abril 2015.
- CAPUA, R. et al. **Um algoritmo heurístico para o problema de bin packing com conflitos**. SBPO - Simpósio Brasileiro de Pesquisa Operacional. [S.l.]: [s.n.]. 2015. p. 1-10.
- CARVALHO, J. M. V. D. LP models for bin packing and cutting stock problems. **European Journal of Operational Research**, v. 141, n. 2, p. 253-273, Setembro 2002.
- CHERRI, A. C. et al. The one-dimensional cutting stock problem with usable leftovers - A survey. **European Journal of Operational Research**, v. 236, n. 2, p. 395-402, Julho 2014.
- CHU, C.; ANTONIO, J. Approximation Algorithms to Solve Real-Life Multicriteria Cutting Stock Problems. **Operations Research**, v. 47, n. 4, p. 495-508, Julho 1999.

- COFFMAN JR., E. G.; GARY, M. R.; JOHNSON, D. S. **Approximation algorithms for bin packing: a survey**, in D. Hochbaum (Ed.) *Approximation Algorithms for NP-Hard Problems*. Boston: PWS Publishing, 1.997.
- CSIRIK, J. et al. A self-organizing bin packing heuristic. **Workshop on Algorithm Engineering and Experimentation**, Janeiro 1999. 250-269.
- CSIRIK, J. et al. On the Sum-of-Squares algorithm for bin packing. **Journal of the ACM (JACM)**, v. 53, n. 1, p. 1-65, Janeiro 2006.
- CUI, Y. et al. A heuristic for the one-dimensional cutting stock problem with pattern reduction. **Engineering Manufacture**, v. 222, p. 677-685, Junho 2008.
- CUNHA, C. B. D. et al. **Heurísticas para o problema de Bin Packing no contexto da distribuição física de produtos**. XL Simpósio Brasileiro de Pesquisa Operacional. [S.I.]: [s.n.]. 2008. p. 712-723.
- DANTZIG, G. B.; WOLFE, P. Decomposition Principle for Linear Programs. **Operations Research**, v. 8, n. 1, p. 101-111, 1960.
- DELORME, M.; IORI, M.; MARTELLO, S. Bin packing and cutting stock problems: Mathematical models and exact algorithms. **European Journal of Operational Research**, v. 255, n. 1, p. 1-20, Novembro 2016.
- DYCKHOFF, H. A typology of cutting and packing problems. **European Journal of Operational Research**, v. 44, n. 2, p. 145-159, Janeiro 1990.
- FALKENAUER, E. A Hybrid Grouping Genetic Algorithm for Bin Packing. **Journal of Heuristics**, v. 2, n. 1, p. 5-30, Junho 1996.
- FEKETE, S. P.; SCHEPERS, J. New classes of fast lower bounds for bin packing problems. **Mathematical Programming**, v. 91, n. 1, p. 11-31, Setembro 2001.
- FLESZAR, K.; HINDI, K. S. New heuristics for one-dimensional bin packing. **Computers & Operations Research**, v. 29, n. 7, p. 821-839, Junho 2002.
- GAU, T.; WÄSCHER, G. CUTGEN1: A problem generator for the Standard One-dimensional Cutting Stock Problem. **European Journal of Operational Research**, v. 84, n. 3, p. 572-579, Agosto 1995.
- GENT, I. P. Heuristic Solution of Open Bin Packing Problems. **Journal of Heuristics**, v. 3, n. 4, p. 299-304, Março 1998.
- GOURGAND, M.; GRANGEON, N.; KLEMENT, N. **An Analogy between Bin Packing Problem and Permutation Problem: A New Encoding Scheme**. *Advances in Production Management Systems. Innovative and Knowledge-Based Production Management in a Global-Local World*. Ajaccio, France: Proceedings, Part I. 2014. p. 572-579.
- GRADISAR, M. et al. A sequential heuristic procedure for one-dimensional cutting. **European Journal of Operational Research**, v. 114, p. 557-568, Fevereiro 1.999.
- GSCHWIND, T.; IRNICH, S. Dual Inequalities for Stabilized Column Generation Revisited. **Journal on Computing**, v. 28, p. 175-194, Fevereiro 2016.
- GUPTA, J. N. D.; HO, J. C. A new heuristic algorithm for the one-dimensional bin packing problem. **Production Planning & Control**, v. 10, n. 6, p. 598-603, Setembro 1999.
- HEMMELMAYR, V.; SCHMID, V.; BLUM, C. Variable neighbourhood search for the variable sized bin packing problem. **Computers & Operations Research**, v. 39, n. 5, p. 1097-1108, Maio 2012.
- HOTO, S. R. et al. **Problemas de corte em duas fases: uma revisão parcial**. Londrina: Semina: Ciências Exatas de Tecnológicas, v. 28, 2007.
- JAHROMI, M. H. et al. Solving an one-dimensional cutting stock problem by simulated annealing and tabu search. **Journal of Industrial Engineering International**, p. 8-24, Dezembro 2012.

- JOHNSON, D. S. Fast Algorithms for Bin Packing. **Journal of Computer and System Sciences**, v. 8, n. 3, p. 272-314, Junho 1.974.
- JÚNIOR, J. A. S.; PINHEIRO, P. R.; THOMAZ, A. C. F. **Otimização das perdas em cortes guilhotinados para bobinas de aço na indústria metalmeccânica**. São João del Rei: Sociedade Brasileira de Pesquisa Operacional, 2004.
- KHURI, S.; SCHÜTZ, M.; HEITKÖTTER, J. **Evolutionary Heuristics for the Bin Packing Problem**. Artificial Neural Nets and Genetic Algorithms. Vienna: Springer. 1.995. p. 285-288.
- KORF, R. E. **A New Algorithm for Optimal Bin Packing**. Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence. Edmonton, Alberta, Canada: [s.n.]. 2002. p. 731-736.
- LAYEB, A.; CHENCHE, S. **A Novel GRASP Algorithm for Solving the Bin Packing Problem**. I.J. Information Engineering and Electronic Business. [S.I.]: [s.n.]. 2012. p. 8-14.
- LEVINE, J.; DUCATELLE, F. Ant Colony Optimisation and Local Search for Bin Packing and Cutting Stock Problems. **Journal of the Operational Research Society**, v. 7, n. 55, p. 705-716, Março 2003. ISSN 0160-5682.
- LIANG, K.-H. et al. A new evolutionary approach to cutting stock problems with and without contiguity. **Computers & Operations Research**, v. 29, n. 12, p. 1641-1659, Outubro 2002.
- LOH, K.-H.; GOLDEN, B.; WASIL, E. Solving the one-dimensional bin packing problem with a weight annealing heuristic. **Computers & Operations Research**, v. 35, n. 7, p. 2283-2291, Julho 2008.
- MARTEL, C. U. A linear time bin-packing algorithm. **Operations Research Letters**, v. 4, n. 4, p. 189-192, Dezembro 1985.
- MARTELLO, S.; TOTH, P. Lower Bounds and Reduction Procedures for the Bin Packing Problem. **Discrete Applied Mathematics**, v. 28, n. 1, p. 59-70, Julho 1990.
- MUNTEAN, O. **Genetically Designed Heuristics for the Bin Packing Problem**. Proceedings of the 9th annual conference companion on Genetic and evolutionary computation. London, United Kingdom : ACM. 2007. p. 2869-2872.
- PARK, M.-W.; KIM, Y.-D. A Systematic Procedure for Setting Parameters in Simulated Annealing Algorithms. **Computers & Operations Research**, v. 25, n. 3, p. 207-217, 1998.
- RAO, R. L.; IYENGAR, S. S. Bin-Packing by Simulated Annealing. **Computers & Mathematics with Applications**, v. 27, n. 5, p. 71-82, Maio 1993.
- RIETZ, J.; SCHEITHAUER, G.; TERNO, J. Families of non-IRUP instances of the one-dimensional cutting stock problem. **Discrete Applied Mathematics**, v. 121, p. 229-245, 2002.
- SCHEITHAUER, G. et al. Solving one-dimensional cutting stock problems exactly with a cutting plane algorithm. **Journal of the Operational Research Society**, n. 52, 2001.
- SCHOENFIELD, J. E. **Fast exact solution of open bin packing problems without linear programming**. US Army Space and Missile Defense Command. Huntsville, Alabama, USA. 2002.
- SCHOLL, A.; KLEIN, R.; JÜRGENS, C. BISON: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem. **Computers & Operations Research**, v. 24, n. 7, p. 627-645, 1997.

- SCHREIBER, E. L.; KORF, R. E. **Improved Bin Completion for Optimal Bin Packing and Number Partitioning**. Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence. Beijing, China: [s.n.]. 2013. p. 651-658.
- SCHWERIN, P.; WÄSCHER, G. The Bin-Packing Problem: A Problem Generator and Some Numerical Experiments with FFD Packing and MTP. **Transports & Operations Research**, v. 4, n. 5/6, p. 377-389, 1997.
- SIM, K.; HART, E. **Generating single and multiple cooperative heuristics for the one dimensional bin packing problem using a single node genetic programming island model**. Proceedings of the 15th annual conference on Genetic and evolutionary computation. Amsterdam, The Netherlands: Proceeding GECCO 13. 2013. p. 1549-1556.
- SIM, K.; HART, E.; PAECHTER, B. **A Hyper-Heuristic Classifier for One Dimensional Bin Packing Problems: Improving Classification Accuracy by Attribute Evolution**. International Conference on Parallel Problem Solving from Nature. [S.l.]: [s.n.]. 2012. p. 348-357.
- SONUC, E.; SEN, B.; BAYIR, S. **Solving Bin Packing Problem using Simulated Annealing**. Proceedings of 65th ISERD International Conference. Meca, Saudi Arabia: [s.n.]. 2017. p. 83-85.
- STAWOWY, A. Evolutionary based heuristic for bin packing problem. **Computers & Industrial Engineering**, p. 465-474, 2008.
- SUÁREZ, C. D. T.; GONZLEZ, E. P.; RENDÓN, M. V. **A heuristic algorithm for the offline one-dimensional bin packing problem inspired by the point Jacobi matrix iterative method**. MICAI '06. Fifth Mexican International Conference. [S.l.]: [s.n.]. 2006.
- TRKMAN, P.; GRADISAR, M. One-dimensional cutting stock optimization in consecutive time periods. **European Journal of Operational Research**, v. 179, n. 2, p. 291-301, Junho 2007.
- WAGNER, B. J. A genetic algorithm solution for one-dimensional bundled stock cutting. **European Journal of Operational Research**, v. 117, n. 2, p. 368-381, Setembro 1999.
- WÄSCHER, G.; GAU, T. Heuristics for the Integer One-dimensional Cutting Stock Problem: a computational study. **OR Spektrum**, n. 18, p. 131-144, 1996.
- WÄSCHER, G.; HAUßNER, H.; SCHUMANN, H. An Improved typology of cutting and packing problems, v. 183, n. 3, p. 1109–1130, Dezembro 2007.
- YANG, Y.; CUI, Y. A heuristic for one-dimensional cutting stock problem with usable leftover. **European Journal of Operational Research**, v. 204, n. 2, p. 245-250, Julho 2010.
- YESODHA, R.; AMUDHA, T. A comparative study on heuristic procedures to solve bin packing problems. **International Journal in Foundations of Computer Science & Techn**, v. 2, n. 6, p. 37-49, Novembro 2012.

APÊNDICE A – Fórmulas auxiliares

Nesta seção estão listadas as fórmulas auxiliares utilizadas na demonstração das relações matemáticas de melhoria da função objetivo revisada dos problemas de empacotamento.

$$a^2 - b^2 = (a - b)(a + b)$$

$$(a + b)^2 = a^2 + 2ab + b^2$$

$$(a - b)^2 = a^2 - 2ab + b^2$$

$$(a + b)^3 = a^3 + 3a^2b + 3ab^2 + b^3$$

$$(a - b)^3 = a^3 - 3a^2b + 3ab^2 - b^3$$

$$(a + b)^4 = a^4 + 4a^3b + 6a^2b^2 + 4ab^3 + b^4$$

$$(a - b)^4 = a^4 - 4a^3b + 6a^2b^2 - 4ab^3 + b^4$$

ANEXO A – Análise dos parâmetros do *simulated annealing*

Pode-se avaliar a estimativa para os parâmetros de temperatura do *simulated annealing*, válidas somente para os problemas de empacotamento unidimensional, usando como exemplo a instância N2C1W1_A do conjunto Scholl1 sem limite no número máximo de iterações sem melhorias.

Figura 35 – Probabilidade de aceitar soluções piores com $\sigma = 0,95$ e parâmetros da primeira estimativa

Figura 36 – Probabilidade de aceitar soluções piores com $\sigma = 0,96$ e parâmetros da primeira estimativa

Registrando o número de soluções piores aceitas pelo critério de Metropolis obtém-se o gráfico das Figura 35, Figura 36, Figura 37, Figura 38 e Figura 39 para cada valor de σ (respectivamente 0,95, 0,96, 0,97, 0,98 e 0,99):

Figura 37 – Probabilidade de aceitar soluções piores com $\sigma = 0,97$ e parâmetros da primeira estimativa

Figura 38 – Probabilidade de aceitar soluções piores com $\sigma = 0,98$ e parâmetros da primeira estimativa

Figura 39 – Probabilidade de aceitar soluções piores com $\sigma = 0,99$ e parâmetros da primeira estimativa

Outra análise realizada foi o número de soluções avaliadas e tempo de processamento em função do parâmetro σ utilizado no *simulated annealing*. Os resultados indicaram que uma boa escolha para σ é 0,97. Para valores acima de 0,97 o número de soluções avaliadas cresce muito, bem como o tempo de processamento. Por outro lado, para valores abaixo de 0,97, embora o tempo de processamento seja muito menor, o número de soluções avaliadas também é muito menor. Reduzindo também a probabilidade de encontrar soluções ótimas.

Os resultados obtidos para o tempo de processamento e número de soluções avaliadas no teste com a instância N2C1W1_A do conjunto Scholl1 são apresentados no gráfico da Figura 40.

Nos testes com a instância N2C1W1_A do conjunto Scholl1 com a segunda agenda de parâmetros descrita na seção 3.2.3, foram obtidos gráficos de probabilidade de aceitar soluções piores em função da temperatura mais próximos do esperado na teoria do *simulated annealing* para todos os valores de σ testados. Comparando os gráficos da Figura 35, Figura 36, Figura 37, Figura 38 e Figura 39, respectivamente com os da Figura 41, Figura 42, Figura 43, Figura 44 e Figura 45, verifica-se que as

probabilidades de aceitar soluções piores a temperaturas altas são muito maiores na segunda agenda que na primeira, fazendo a busca ser mais aleatória a temperaturas altas na segunda agenda de parâmetros, permitindo maior diversificação das soluções nesta agenda.

Por outro lado, a estimativa para a temperatura final mostrou-se adequada com baixa probabilidade de aceitação de soluções piores a baixas temperaturas, permitindo a intensificação característica do *simulated annealing* a baixas temperaturas.

Na Figura 46 é apresentado o gráfico do número de soluções avaliadas e tempo de processamento em função do parâmetro de resfriamento σ para a segunda agenda de parâmetros.

Figura 40 – Número de soluções avaliadas e tempo de processamento em função de σ e parâmetros da primeira estimativa

Figura 41 – Probabilidade de aceitar soluções piores com $\sigma = 0,95$ e parâmetros da segunda estimativa

Figura 42 – Probabilidade de aceitar soluções piores com $\sigma = 0,96$ e parâmetros da segunda estimativa

Figura 43 – Probabilidade de aceitar soluções piores com $\sigma = 0,97$ e parâmetros da segunda estimativa

Figura 44 – Probabilidade de aceitar soluções piores com $\sigma = 0,98$ e parâmetros da segunda estimativa

Figura 45 – Probabilidade de aceitar soluções piores com $\sigma = 0,99$ e parâmetros da segunda estimativa

Figura 46 – Número de soluções avaliadas e tempo de processamento em função de σ e parâmetros da segunda estimativa

ÍNDICE

BFD, 13, 32, 36, 45, 48, 49, 67, 83, 99
Bin Packing Problem, 13
 BISON, 33, 34
Cutting Stock Problem, 13
 DS, 83, 84
 FFD, 13, 25, 32, 34, 36, 45, 46, 47, 62,
 67, 83, 99
 LFD, 13, 47, 48, 83, 99
 MBS, 13, 33, 44, 45, 46, 67, 83, 84, 99
 MBS', 83, 99
 MBSR, 46, 83, 99
 MTP, 32
 MTPCS, 32, 34

Next Fit Decreasing, 13, 50, 51
 NFD, 13, 50, 51, 67, 83, 99
 PMBS, 33, 34
 SA, 13, 34, 66, 67, 72, 83, 84, 99
Simulated Annealing, 8, 13, 34, 66, 72,
 84
Sum-of-Squares, 13, 42, 43
Variable Neighborhood Search, 33
 WABP, 34
 Weighted Annealing, 33
 WFD, 13, 49, 50, 83, 99
Worst Fit Decreasing, 14, 49, 50