PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO E SISTEMAS

RICARDO DE ALMEIDA

PROPOSTA DE UM MÉTODO META-HEURÍSTICO HÍBRIDO PARA RESOLUÇÃO DE PROBLEMAS DO TIPO *BIN PACKING*.

CURITIBA
2014

RICARDO DE ALMEIDA

PROPOSTA DE UM MÉTODO META-HEURÍSTICO HÍBRIDO PARA RESOLUÇÃO DE PROBLEMAS DO TIPO *BIN PACKING*.

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia de Produção e Sistemas, área de concentração: Gerência de Produção e Logística, da Pontifícia Universidade Católica do Paraná, como requisito parcial à obtenção do título de Mestre em Engenharia de Produção.

Orientadora: Profa. Dr.ª Maria Teresinha Arns Steiner.

CURITIBA

2014

RICARDO DE ALMEIDA

PROPOSTA DE UM MÉTODO META-HEURÍSTICO HÍBRIDO PARA RESOLUÇÃO DE PROBLEMAS DO TIPO *BIN PACKING*.

Este trabalho, que é requisito para a qualificação no Programa de Pós-Graduação em Engenharia de Produção e Sistemas, área de concentração: Gerência de Produção e Logística, da Pontifícia Universidade Católica do Paraná, foi avaliado pela seguinte banca examinadora:

COMISSÃO EXAMINADORA

_____

Orientadora

Prof. Dr.ª Maria Teresinha Arns Steiner

_____

1º Membro Avaliador

Prof. Dr. Celso Carnieri

_____

2º Membro Avaliador

Prof. Dr. Cassius Tadeu Scarpin

_____

3º Membro Avaliador

Prof. Dr. Luciano Antonio Mendes

Curitiba, ____ de _____ de 2014.

AGRADECIMENTOS

A Deus, primeiramente, pelo dom da vida e por permitir que este trabalho fosse realizado.

À minha orientadora, Maria Teresinha Arns Steiner, grande exemplo de profissional, pela dedicação, atenção, paciência e amizade. Uma grande pessoa sempre disposta a ajudar a todos em todas as áreas.

Aos meus pais, João e Sueli, por toda dedicação na minha educação, pelo apoio em todos meus projetos e por serem as pessoas com quem, juntamente com meu irmão Fabiano, sei que sempre posso contar nas horas difíceis.

À todos os professores e funcionários do programa de Pós-Graduação da Pontifícia Universidade Católica do Paraná, pela dedicação e orientação, sendo fundamental para a qualidade do programa.

À todos os colegas de classe que de alguma forma colaboraram para realização deste trabalho, seja com apoio, ideias ou simplesmente inspiração.

Ao Programa de Pós-Graduação em Engenharia de Produção e Sistemas da Pontifícia Universidade Católica do Paraná por todo apoio e pela concessão da bolsa de estudos que foi determinante para conclusão desta pesquisa.

Aos membros da banca, Prof. Dr. Cassius Tadeu Scarpin, Prof. Dr. Celso Carnieri e Prof. Dr. Luciano Antonio Mendes pelas importantes e valiosas sugestões que elevaram a qualidade deste trabalho e servirão de lição para trabalhos futuros.

À todos os amigos e familiares que de alguma forma contribuíram e apoiaram no decorrer deste desafio.

RESUMO

A otimização de recursos e atividades tem se tornado um requisito básico para as empresas que pretendem ser competitivas no mercado. Muitos dos problemas que fazem parte do cotidiano das empresas e que são passíveis de otimização recaem nas classes de problemas de otimização combinatória do tipo NP-difícil, entre eles o Problema de Corte e Empacotamento (PCE). PCEs são bastante comuns em diversos tipos de indústria e o adequado tratamento deste tipo de problema pode trazer como principais benefícios a economia de matéria prima (corte) ou otimização de espaços (empacotamento), além de outros decorrentes destes principais, com potencial para significativas reduções em custos. Devido à complexidade deste tipo de problema, a resolução de casos reais exige métodos robustos e eficientes, capazes de fornecer repostas ótimas, ou muito próximas a esta, em tempo reduzido. Neste sentido, várias pesquisas têm sido desenvolvidas na busca de métodos cada vez mais eficazes para resolução de PCEs. Neste trabalho são abordados métodos para resolução de PCE unidimensional, são eles: métodos de aproximação (*First-Fit*, *Best-Fit* e *Worst-Fit*) e suas variações; variações do método *Minimum Bin Slack* de Gupta e Ho (1999); métodos baseados em busca local (*Variable Neighbourhood Search*, *Weight Annealing* e *Augmented Neural Networks*), que são analisados em diversas configurações, além do método híbrido *Perturbation* MBS'+VNS de Fleszar e Hindi (2002). Os algoritmos são avaliados através da aplicação em problemas encontrados na literatura, num total de 1.360 casos. Os problemas resolvidos neste trabalho fazem parte de um caso particular do PCE, o problema do tipo *Bin Packing*, caracterizado por forte heterogeneidade de itens e fraca heterogeneidade de objetos. Também é proposto o método MBS'_Pert_WABP, um método meta-heurístico híbrido que foi capaz de superar todos os métodos analisados neste trabalho em termos de qualidade de solução, diminuindo o erro total em 54,6% em relação ao segundo melhor método, sendo que, em dois casos, o novo método foi capaz de encontrar respostas melhores que a melhor solução conhecida na literatura.

Palavras-chave: Problemas de corte e empacotamento, *Bin packing*, heurísticas, meta-heurísticas.

ABSTRACT

The optimization of resources and activities is a basic requirement for the companies that intend to be competitive in the market. Many of the problems that arise in the daily life of companies and could be optimized fall into the class of NP-hard combinatorial optimization problems, among them, Cutting and Packing Problems (CPP). CPPs are very common in various types of industries and its proper treatment could result in many benefits, as savings in raw material (cutting) and optimization of physical spaces (packing), resulting in potential cost reduction. Given the complexity of this kind of problems, the resolution of problems found in the real world requires efficient and robust methods that are able to find optimal or near to optimal solutions in a short period of time. In this sense, many researches have been developed in search of more effective methods. In this work methods for resolution of one-dimensional CPPs are analyzed: approximation methods (First-Fit, Best-Fit and Worst-Fit) and their variations; variations of Minimum Bin Slack of Gupta and Ho (1999); methods based on local search (Variable Neighbourhood Search, Weight Annealing and Augmented Neural Networks), which are tested in different configurations; and the hybrid method Perturbation MBS' + VNS proposed by Fleszar and Hindi (2002). The algorithms are evaluated through the application in 1.360 problems available in literature. The instances solved in this work are a particular case of CPP, the Bin Packing Problem, characterized by strong heterogeneity of items and weak heterogeneity of objects. Also the MBS'_Pert_WAPB, a new hybrid meta-heuristic method, that was able to overcome all other methods evaluated in this work, is proposed. The new method reduced 54,6% the total error compared to the second best method and was able to improve two best known solutions.

Keywords: Cutting and Packing Problems, Bin Packing, heuristics, meta-heuristics.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

AG – Algoritmo Genético

AGG - Algoritmo de Grupamento Genético

AGGH - Algoritmo de Grupamento Genético Híbrido

AugNN – *Augmented Neural Networks*

B&B – *Branch & Bound*

BF – *Best-Fit*

BFD – *Best-Fit-Decreasing*

BP – *Bin Packing*

BT – Busca Tabu

DA – *Deterministic Annealing*

FF – *First-Fit*

FFD – *First-Fit-Decreasing*

LI – Limite Inferior

MBS – *Minimum Bin Slack*

NF – *Next-Fit*

NFD – *Next-Fit-Decreasing*

PCE – Problema de Corte e Empacotamento

PL – Programação Linear

PLI - Programação Linear Inteira

RNA – Rede Neural Artificial

VNS – *Variable Neighbourhood Search*

SA – *Simulated Annealing*

WA – *Weight Annealing*

WABP – *Weight Annealing Bin Packing*

WF – *Worst-Fit*

WFD – *Worst-Fit-Decreasing*

SUMÁRIO

1 INTRODUÇÃO

A dinâmica dos mercados exige das empresas respostas cada vez mais rápidas e eficientes às demandas que se lhe fazem face. Muitas empresas enfrentam *tradeoffs* entre *responsividade*, ou seja, capacidade de se adaptar rapidamente às mudanças do mercado, e eficiência, que está associada a custos reduzidos, conseguidos com gerenciamento eficaz de estoque, ganhos de escala, economia em compras, dentre outros. Neste contexto, o Problema de Corte e Empacotamento (PCE) exerce importante influência nos processos onde é encontrado. De forma simples, o problema de corte consiste em, dado um objeto em estoque, obter a partir deste objeto, itens menores para atendimento de uma determinada demanda. No caso do empacotamento, o objetivo é alocar itens menores em um objeto maior, sempre levando em consideração a otimização de um ou mais critérios de desempenho.

PCEs são encontrados sob diversas formas e em diversos ramos da indústria e serviços. Na indústria metalúrgica, o gerenciamento adequado do corte de perfis e chapas metálicas pode representar uma importante fonte de economia de matérias-primas, além de redução de estoques em processo e necessidade de manuseio inadequado. Também a indústria moveleira se beneficia de um melhor aproveitamento de matérias-primas, com consequente redução de resíduos, reduzindo custos tanto em insumos como no tratamento dos resíduos. No setor logístico, o carregamento eficiente de contêineres pode trazer benefícios de redução de custo de transporte. No setor informático, o crescimento exponencial da produção e consequente armazenamento de informação, exige gerenciamento adequado da utilização de servidores, facilitando o acesso à informação e reduzindo energia.

Os PCEs são problemas de otimização combinatória classificados como NP-difíceis. Como esta classe de problemas é difícil de ser resolvida de forma ótima em tempo computacional razoável, diversos métodos heurísticos e meta-heurísticos tem sido desenvolvidos para tratar problemas reais de grande porte (AYOB *et al*., 2013).

A realização deste trabalho surge da motivação de buscar métodos mais eficazes e eficientes para resolução deste tipo de problema, haja vista sua importância para a indústria e a dificuldade encontrada para resolução, principalmente de problemas reais de grande porte.

1.1 OBJETIVOS

Os objetivos almejados neste trabalho foram divididos em duas categorias, geral e específicos.

1.1.1 Objetivo Geral

O objetivo principal desta pesquisa é descrever alguns métodos utilizados para resolução de PCEs, mais especificamente problemas do tipo *Bin Packing* (BP), além de propor novo método híbrido para resolução deste tipo de problema e avaliar o desempenho dos métodos analisados neste trabalho sob os critérios de minimização de perdas e velocidade de obtenção de resposta.

1.1.2 Objetivos Específicos

Como decorrência do objetivo geral, pretende-se alcançar os seguintes objetivos específicos:

a) Implementar os algoritmos dos métodos de aproximação *First-Fit* (FF); *First-Fit-Decreasing* (FFD); *Best-Fit* (BF); *Best-Fit-Decreasing* (BFD); *Worst-Fit* (WF) e *Worst-Fit-Decreasing* (WFD).
b) Implementar os algoritmos dos métodos baseados em *Minimum Bin Slack* (MBS): MBS; MBS' e *Perturbation* MBS'. Testar variações de solução inicial para o método *Perturbation* MBS'.
c) Implementar os algoritmos dos métodos baseados em busca local: *Augmented Neural Networks* (AugNN); *Variable Neighbourhood Search* (VNS) e *Weight Annealing Bin Packing* (WABP). Testar variações de solução inicial para cada um dos métodos.
d) Implementar o algoritmo do método híbrido *Perturbation* MBS' + VNS.
e) Propor novo método híbrido para resolução de problemas do tipo BP, no qual a solução inicial é construída pelo método MBS' (*Minimum Bin Slack*), posteriormente perturbada pelo método *Perturbation* MBS' e refinada pelo método WABP.
f) Aplicar os métodos de resolução em problemas da literatura disponíveis em *OR-Library* (https://files.nyu.edu/jeb21/public/jeb/info.html);
g) Comparar e analisar os resultados obtidos pelos diferentes métodos.

1.2 LIMITAÇÕES DO TRABALHO

Os métodos abordados neste trabalho são específicos para resolução de problemas do tipo *Bin Packing* unidimensional. Tais métodos podem ser adaptados para resolução de outras variações dos PCEs além do tipo *Bin Packing*, como o problema de Corte de Estoque, por exemplo, entretanto não são aplicáveis a problemas com duas ou mais dimensões. Aspectos práticos característicos dos processos industriais como, limitação de máquinas ou tipos de processos de corte, por exemplo, também não são considerados.

1.3 ESTRUTURA DO TRABALHO

Este trabalho foi estruturado da seguinte forma: no capítulo 2 é apresentada a estrutura do PCE, a classificação e nomenclatura utilizada como referência e os principais modelos matemáticos. No capítulo 3 é apresentada, através de trabalhos relacionados aos PCEs, a evolução da pesquisa nesta área com alguns exemplos de aplicação e métodos de resolução. No capítulo 4, são detalhados os métodos de resolução que serão estudados neste trabalho.

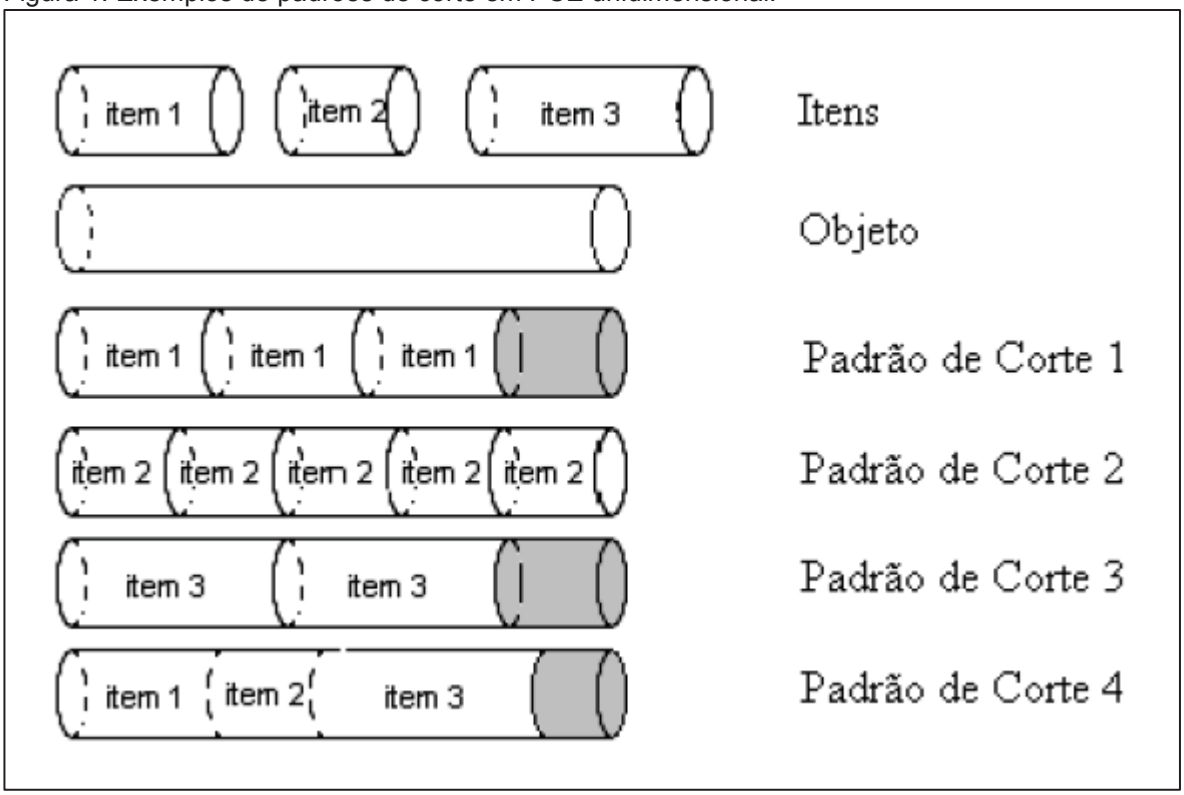
O capítulo 5 explica a metodologia de implementação computacional e a estrutura dos problemas nos quais os métodos são testados. O capítulo 6 mostra os resultados computacionais de cada método, apresentando as devidas comparações. Também é detalhado um método híbrido existente na literatura e uma nova proposta de hibridização com seus respectivos resultados.

Por fim, no capítulo 7, encontram-se as conclusões e direcionamentos futuros.

2 O PROBLEMA DE CORTE E EMPACOTAMENTO

Os problemas de corte e empacotamento e consistem em, dado um conjunto de objetos grandes e um conjunto de itens pequenos, agrupar os itens, formando subconjuntos que serão então arranjados nos objetos grandes. Os itens podem ser peças para atendimento de uma demanda ou caixas a serem carregadas em um contêiner, por exemplo, enquanto o objeto trata-se da matéria-prima da qual serão retirados os itens, ou o contêiner no qual serão carregadas as caixas. Os subconjuntos de itens formam padrões de corte, ou padrões de empacotamento. A figura 1 ilustra um caso unidimensional onde se observam itens, objetos e padrões de corte, sendo que a área em cinza são as sobras de corte.

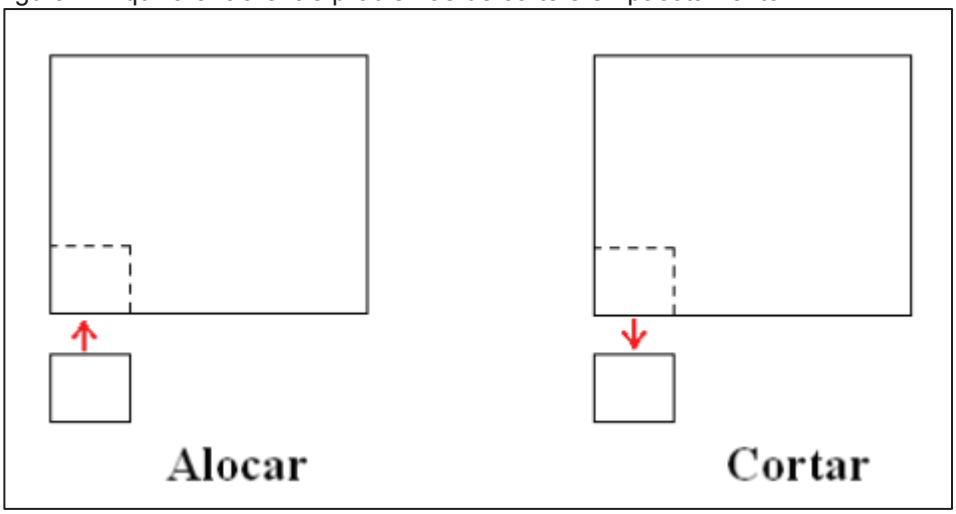
Figura 1: Exemplos de padrões de corte em PCE unidimensional.



Fonte: Mosqueira (2007).

Problemas de corte e empacotamento possuem estrutura idêntica, sendo que, enquanto o primeiro consiste em cortar itens a partir de objetos grandes o segundo trata de alocar itens em objetos maiores. A figura 2 mostra a equivalência entre os problemas de corte e empacotamento.

Figura 2: Equivalência entre problemas de corte e empacotamento.



Fonte: Mosqueira (2007).

Uma função objetivo deve ser otimizada e o agrupamento de itens deve ser feito de forma que não viole as condições geométricas do problema, ou seja, não haja sobreposição de itens e que os subconjuntos de itens não excedam as dimensões do objeto ao qual está sendo designado (Whäscher *et al*, 2007). Os PCEs podem ser definidos em 1, 2 ou 3 dimensões geométricas. Mosqueira (2007) destaca que as dimensões consideradas nos PCEs podem ser espaciais, que estão relacionadas às dimensões geométricas, ou não espaciais, relacionadas a outras restrições, como limitação de carga admitida por um objeto ou valor dos itens a serem cortados/empacotados no problema. Também é importante que sejam observadas condições técnicas pertinentes a cada problema como: perda de material devido à espessura de serra de corte, número de facas no corte de bobinas de papel, limite de peças por objeto, dentre outros. É um problema de grande relevância na indústria, visto que diversos processos necessitam de transformação de objetos em itens menores, de forma a suprir uma determinada demanda.

2.1 CLASSIFICAÇÃO E NOMENCLATURA

Tanto os problemas de corte como os de empacotamento são encontrados de diversas formas na literatura, desde *Trim Problem* mostrado por Eisemann (1957), até alocação de memória, passando por *bin packing*, problema da mochila, carregamento de veículos, particionamento, grupamento, entre outros. Também são várias as áreas que tratam de problemas desta natureza: Matemática; Ciência da Computação; Pesquisa Operacional; Logística e Engenharia Industrial; estão entre

algumas citadas por Dyckhoff (1990), a quem é atribuída a autoria do primeiro trabalho com objetivo de integrar e classificar os diversos problemas e nomenclaturas existentes relacionados à estrutura básica de PCEs, de forma a criar uma tipologia para referência.

Algumas deficiências na tipologia proposta por Dyckhoff (1990), principalmente devido a desenvolvimentos subsequentes, a impediram de ser mais amplamente aceita e inspiraram o trabalho de Wäscher *et al*. (2007), que propuseram uma metodologia de classificação mais abrangente, inspirada nas ideias originais de Dyckhoff (1990). A tipologia criada por Wäscher *et al*. (2007), comumente utilizada em trabalhos recentes, será explicada a seguir e será utilizada como referência no decorrer deste trabalho.

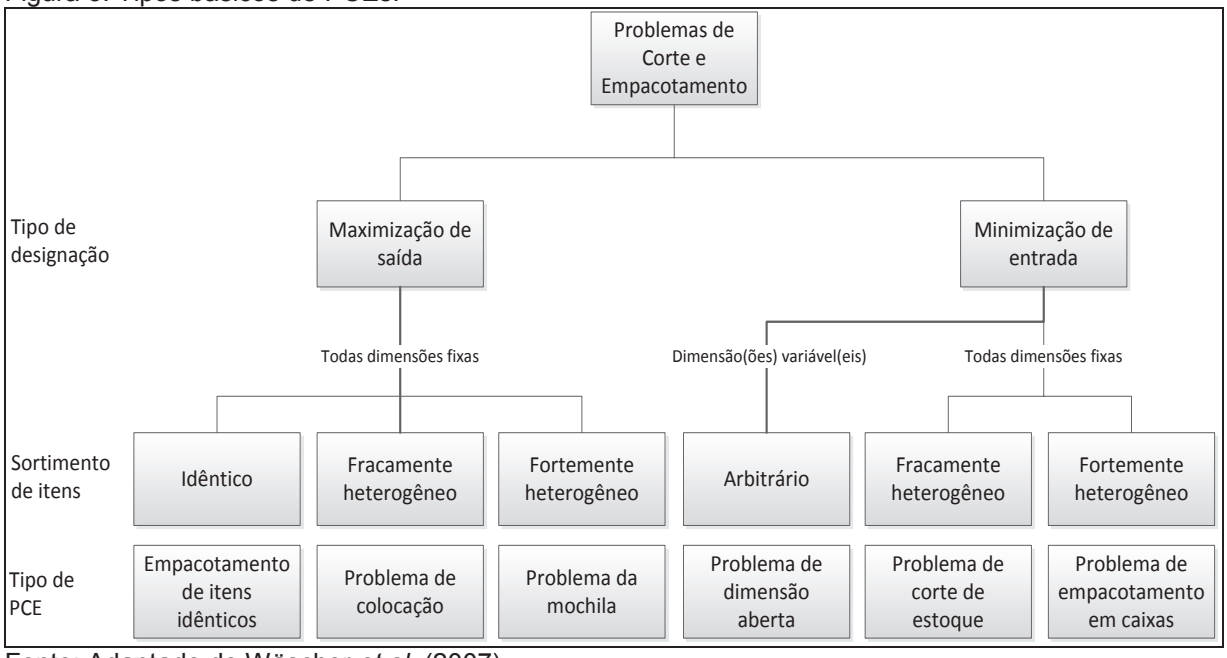
Para caracterizar os tipos de problemas variantes dos PCEs, Wäscher *et al*. (2007) utilizaram cinco critérios: *tipo de designação*; *sortimento dos itens pequenos*; *sortimento dos objetos grandes*; *dimensionalidade* e; *forma dos itens pequenos*. Os critérios *tipo de designação* e *sortimento de itens pequenos*, que quando combinados descrevem os tipos básicos de problemas propostos por Wäscher *et al*. (2007), são descritos a seguir:

- **Tipo de designação**: pode ser *maximização das saídas* ou *minimização das entradas*. No primeiro, um conjunto de itens pequenos deve ser designado a um dado conjunto de objetos grandes. Todos os objetos devem ser utilizados e sua disponibilidade não é suficiente para acomodar todos os itens. Desta forma, o subconjunto de itens com **máximo** valor deve ser designado. No caso de *minimização de entrada*, também um conjunto de itens pequenos deve ser designado a um conjunto de objetos grandes, entretanto, a quantidade disponível de objetos é suficiente para acomodar todos os itens. Desta forma, um subconjunto de objetos de **mínimo** valor deve ser escolhido. O valor a ser maximizado ou minimizado depende de cada problema específico e pode estar relacionado a custo, lucro ou quantidade de matéria-prima, por exemplo.

- **Sortimento de itens pequenos**: São classificados em três casos. *Itens pequenos idênticos*, onde os itens, relativamente às dimensões consideradas no problema, possuem mesma forma e tamanho. No segundo caso têm-se itens com *sortimento fracamente heterogêneo* onde os itens pequenos podem ser agrupados em algumas poucas classes de itens idênticos. No terceiro caso muito poucos itens possuem dimensão e forma idênticas, caracterizando *sortimento fortemente heterogêneo*.

A figura 3 mostra a correspondência entre a combinação das características *tipo de designação* e *sortimento de itens pequenos*, além dos problemas básicos decorrentes: Problema de empacotamento de itens idênticos (*Identical Item Packing Problem*), problema de colocação (*Placement Problem*), problema da mochila (*Knapsack Problem*), Problema de dimensão aberta (*Open Dimension Problem*), Problema de corte de estoque (*Cutting Stock Problem*) e problema de empacotamento em caixas (*Bin Packing Problem*).

Figura 3: Tipos básicos de PCEs.



Fonte: Adaptado de Wäscher *et al*. (2007).

O critério *sortimento de objetos grandes*, descrito a seguir, tem como objetivo estruturar os problemas básicos em problemas intermediários.

- **Sortimento de objetos grandes**: Nesta categoria, têm-se duas classificações, *um objeto grande* e *vários objetos grandes*. *Um objeto grande* consiste na utilização de apenas um objeto e dentro desta categoria, o objeto pode ter todas suas dimensões fixas, ou ter uma ou mais dimensões variáveis, como por exemplo, na utilização de rolos de tecido para corte, onde a largura é fixa e o comprimento utilizado deve ser minimizado. A categoria *vários objetos grandes*, considera todas as dimensões fixas e é subdividida em: objetos *idênticos*; objetos com sortimento *fracamente heterogêneo*, ou múltiplos tamanhos, e; objetos com sortimento *fortemente heterogêneo*, também chamado de residual.

A combinação dos critérios sortimento de itens pequenos e sortimento de objetos grandes cria um panorama para problemas intermediários de maximização de saídas, apresentado na figura 4, e um panorama de problemas intermediários de minimização de entradas, mostrado na figura 5.

Figura 4: Panorama de problemas intermediários para o critério de maximização das saídas.

Característica dos objetos \ Sortimento de itens		Idêntico	Fracamente heterogêneo	Fortemente heterogêneo
Todas as dimensões fixas	Um objeto grande	Empacotamento de itens idênticos (*Identical Item Packing Problem*) IIPP	Problema de colocação em um objeto (*Single Large Object Placement Problem*) SLOPP	Problema de uma mochila (*Single Knapsack Problem*) SKP
	Idêntico		Problema de colocação em múltiplos objetos idênticos (*Multiple Identical Large Object Placement Problem*) MILOPP	Problema de múltiplas mochilas idênticas (*Multiple Identical Knapsack Problem*) MIKP
	Heterogêneo		Problema de colocação com múltiplos objetos heterogêneos (*Multiple Heterogeneous Large Object Placement Problem*) MHLOPP	Problema de múltiplas mochilas heterogêneas (*Multiple Heterogeneous Knapsack Problem*) MHKP

Fonte: Adaptado de Wäscher *et al*. (2007)

Figura 5: Panorama de problemas intermediários para o critério de minimização das entradas.

Característica dos objetos \ Sortimento de itens		Fracamente heterogêneo	Fortemente heterogêneo
Todas dimensões fixas	Idêntico	Problema de colocação em um objeto (Single Large Object Placement Problem) SLOPP	Problema de uma mochila (Single Knapsack Problem) SKP
	Fracamente heterogêneo	Problema de colocação em múltiplos objetos idênticos (Multiple Identical Large Object Placement Problem) MILOPP	Problema de múltiplas mochilas idênticas (Multiple Identical Knapsack Problem) MIKP
	Fortemente heterogêneo	Problema de colocação com múltiplos objetos heterogêneos (Multiple Heterogeneous Large Object Placement Problem) MHLOPP	Problema de múltiplas mochilas heterogêneas (Multiple Heterogeneous Knapsack Problem) MHKP
Um objeto grande com dimensão(ões) variável(eis)		Problema de dimensão aberta (Open Dimension Problem) ODP	

Fonte: Adaptado de Wäscher *et al*. (2007)

O refinamento dos tipos de problemas é dado pelos critérios *dimensionalidade* e *forma dos itens*, detalhados a seguir:

- **Dimensionalidade**: distingue os problemas em 1, 2 ou 3 dimensões. Eventualmente mais de 3 dimensões (n > 3) podem ser consideradas, como por exemplo, trabalha com itens de 3 dimensões geométricas com restrição de peso. Neste caso, os problemas são considerados como variantes.

- **Forma dos itens pequenos**: Para problemas de dimensão 2 ou 3, os itens são classificados em *regulares* e *irregulares*, sendo que itens *regulares* de 2 dimensões podem ser ainda divididos em itens *retangulares* e *circulares*.

Com isto, a classificação proposta por Wäscher *et al*. (2007) é dada pelo tipo de problema intermediário antecedido pelos critérios de *dimensionalidade* e *forma dos itens*, quando aplicável, formando uma nomenclatura com a seguinte estrutura: **{1,2,3}-dimensão(ões) {retangular, circular,...,irregular} {problema intermediário}.**

2.2 FORMULAÇÃO MATEMÁTICA

Carvalho (2002) no trabalho "*LP models for bin packing and cutting stock problems*", explora vários modelos matemáticos de PL utilizados na resolução de problemas de corte de estoque e BP. A solução ótima de modelos matemáticos de PLI pode ser encontrada, através de *softwares* matemáticos em tempo computacional razoável, nos problemas de pequeno porte. Problemas maiores, principalmente problemas com sortimento de itens fracamente heterogêneo, podem ser resolvidos através da relaxação do modelo de PLI, gerando soluções fracionadas que são posteriormente tratadas em termos de arredondamento através de procedimentos heurísticos. Em problemas do tipo BP, devido principalmente à forte heterogeneidade de itens, as soluções podem se tornar difíceis de serem arredondadas à medida que cresce a quantidade de padrões de corte diferentes na solução. Carvalho (2002) cita, para problemas do tipo Corte de Estoque e BP, os modelos de Kantorovich, Gilmore-Gomory, modelo de um-corte, modelo de posição-indexada e um modelo derivado do Roteamento de Veículos.

O modelo proposto por Kantorovich (1939), que apesar de relativamente antigo ainda é vastamente difundido, tem como objetivo minimizar o número de objetos utilizados para produção dos itens requeridos pela demanda. Tal modelo encontra-se descrito pela função (2.1) e inequações (2.2) e (2.3), a seguir, sendo que em (2.4) e (2.5) tem-se definido o domínio do problema.

$$min \quad \sum_{k=1}^{K} y_k \tag{2.1}$$

$$s.a \quad \sum_{k=1}^{K} x_{ik} \geq b_{i,} \quad i = 1, 2, \ldots, m, \tag{2.2}$$

$$\sum_{i=1}^{m} w_i x_{ik} \leq W y_k, \quad k = 1, 2, \ldots, K, \tag{2.3}$$

$$y_k \in \{0,1\}, \quad k = 1, 2, \ldots, K, \tag{2.4}$$

$$x_{ik} \in \mathbb{Z}_+, \quad i = 1, 2, \ldots, m, \quad k = 1, 2, \ldots, K. \tag{2.5}$$

Neste modelo, K é um limitante superior da quantidade de objetos necessária, as variáveis y_k assumem o valor “1”, se o objeto k é utilizado e “0”, caso contrário; x_{ik} representa a quantidade de vezes que o item i é atribuído ao objeto k; b_i representa a demanda pelo item i e w_i o seu comprimento. W representa o tamanho do objeto que, neste caso, é considerado de tamanho único. A função (2.1) representa a função objetivo a ser otimizada, (2.2) garante que toda demanda seja atendida, (2.3) garante que o somatório dos comprimentos dos itens atribuídos a um determinado objeto seja menor que o comprimento do objeto; a restrição (2.4) indica se um objeto é utilizado ou não e a restrição (2.5) garante que um item não seja fracionado.

O modelo de Kantorovich pode ser aplicado principalmente em problemas com fraca heterogeneidade de itens, por exemplo, do tipo IIPP, SLOPP, SKP, MILOPP e MIKP, e pode ser adaptado para utilização em problemas com forte heterogeneidade de objetos, através da indexação do comprimento do objeto (W_k), para os comprimentos disponíveis. Entretanto, como a função objetivo (2.1) considera apenas se o objeto é utilizado ou não, o modelo tenderá naturalmente a priorizar os objetos maiores, aos quais pode ser designada maior quantidade de itens. Em muitos casos, como na utilização de sobras de objetos na indústria, por exemplo, pode ser interessante priorizar a utilização destas sobras, esta situação requer que a função objetivo seja adaptada, penalizando a utilização de objetos novos através da consideração de custos ou pesos para cada objeto.

A formulação de Gilmore e Gomory para corte de estoque (Gilmore e Gomory, 1961, 1963) é um modelo de PLI derivado do modelo de Eisemann (1957), que tem como inconveniente a necessidade de enumeração de todos os padrões de corte

factíveis possíveis, sendo que a quantidade de padrões cresce de forma exponencial conforme aumenta a quantidade demandada de itens. No modelo de Gilmore e Gomory ((2.6) a (2.8)), a variável λ^p, que representa a quantidade de padrões de corte, pode ser relaxada alterando a restrição (2.8) de inteiros não negativos para reais não negativos, transformando o modelo de PLI em PL. Desta forma, ao resultado do modelo de PL podem ser aplicados métodos de arredondamento, sendo que nos casos onde a variável λ^p assume valores grandes, o arredondamento pode representar um custo relativamente pequeno, sendo eficiente principalmente em problemas do tipo IIPP, MILOPP e MHLOPP. Para tratar o inconveniente da grande quantidade de padrões, Gilmore e Gomory (1961) propõem um método de geração de colunas com a introdução de um problema auxiliar de programação inteira, neste caso, um *Problema da Mochila* (SLOPP ou SKP). O método consiste basicamente em gerar uma solução inicial factível e, a partir dos custos duais desta solução, resolver o problema auxiliar do tipo *Problema da Mochila*, cuja resolução gera um padrão de corte viável, representando uma coluna do problema mestre no método Simplex Revisado. A inclusão adequada desta coluna gera uma solução melhor ou igual a solução precedente.

$$min \quad \sum_{p \in P} \lambda^p \tag{2.6}$$

$$s.a. \quad \sum_{p \in P} a_i^p \lambda^p \geq b_i, \qquad i = 1, 2, \dots, m, \tag{2.7}$$

$$\lambda^p \in \mathbb{Z}_+, \qquad \forall p \in P \tag{2.8}$$

A função objetivo (2.6) minimiza a quantidade de objetos utilizados, sendo *p* um padrão pertencente a uma coleção *P* de padrões viáveis. Está implícito que os objetos são idênticos, entretanto é possível associar custos a cada objeto, para diferenciar, por exemplo, objetos com tamanhos diferentes ou objetos com acesso mais difícil. Reescrevendo a função (2.6) para incluir os custos, esta assume a forma de (2.9):

$$min \quad \sum_{p \in P} c_p \lambda^p \tag{2.9}$$

O conjunto de restrições representado por (2.7) garante que toda demanda seja atendida sendo a_i^p a quantidade itens do tipo i no padrão p. A restrição (2.8), usualmente relaxada, define que a quantidade de cada padrão p, pertencente ao conjunto P, seja inteira e não negativa.

A geração de padrões é dada pela função (2.10), onde π_i representa os custos duais do problema mestre. As restrições representadas por (2.11) garantem que o padrão seja factível, e a indexação de W permite que se trabalhe com mais de um tipo de comprimento de objeto, e neste caso, um problema auxiliar deve ser resolvido para cada tipo de objeto, sendo o escolhido o que gere maior valor da função objetivo (2.10). A restrição (2.12) determina que a variável a_i^p seja inteira e não negativa, ou seja, frações de item ou quantidades negativas não são aceitas.

$$max \quad \sum_{i=1}^{m} \pi_i \, a_i^p \tag{2.10}$$

$$s.a. \quad \sum_{i=1}^{m} a_i^p w_i \leq W, \qquad i = 1, 2, \ldots, m, \tag{2.11}$$

$$a_i^p \in \mathbb{Z}_+, \qquad i = 1, 2, \ldots, m. \tag{2.12}$$

Dois modelos apresentados por Carvalho (2002) tratam o PCE de forma indexada, ou seja, a informação sobre a posição do item no objeto é relevante: o modelo de *Fluxo em Arco* e o modelo de *Uns Consecutivos*. Ambos os modelos se enquadram em problemas de objetos variados e são extensões dos modelos apresentados por Carvalho (1999) para problemas de objetos idênticos. O primeiro, modelo de *Fluxo em Arco*, tem por objetivo otimizar a função (2.13), respeitando as restrições (2.14), (2.15), (2.16), (2.17) e (2.18).

$$min \quad \sum_{k=1}^{K} W_k z_k \tag{2.13}$$

$$s.a. \quad -\sum_{(d,e)\in A'} x_{de} + \sum_{(e,f)\in A'} x_{ef} = \begin{cases} \sum_{k=1}^{K} z_k & se\ e = 0, \\ -z_k & para\ e = W_k, k = 1,2,\ldots,K, \\ 0 & caso\ contrário, \end{cases} \tag{2.14}$$

$$\sum_{(d,d+w_i)\in A'} x_{d,d+w_i} \geq b_i, i = 1,2,\ldots,m, \tag{2.15}$$

$$z_k \leq B_k, k = 1,2,\ldots,K, \tag{2.16}$$

$$x_{de} \in \mathbb{Z}_+, \qquad \forall (d,e) \in A', \tag{2.17}$$

$$z_k \in \mathbb{Z}_+, \ \ k = 1,2,\ldots,K \tag{2.18}$$

Este modelo parte do agrupamento dos objetos em K classes de diferentes capacidades $W_k, k = 1,2,\ldots,K$, sendo $B_k, k = 1,2,\ldots,K$, a quantidade de objetos em cada classe. Também os itens são agrupados em classes de acordo com seu tamanho ($w_i, i = 1,2,\ldots,m$), sendo $b_i, i = 1,2,\ldots,m$, a quantidade de itens em cada classe. Os itens e objetos são indexados em ordem decrescente de tamanho, sendo $W_{max} = max_k W_k = W_1$, considera-se um grafo $G = (V, A)$ como um conjunto de vértices $V = \{0,1,2,\ldots,W_{max}\}$, e um conjunto de arcos $A = \{(d,e): 0 \leq d < e \leq W_{max} \wedge e - d = w_i \ \forall\ 1 \leq i \leq m\}$, o que significa que existe um arco direcionado entre 2 vértices se existir um item de tamanho correspondente. Consideram-se também arcos adicionais entre $(d, d+1), d = 0,1,\ldots,W_{max} - 1$, que correspondem a espaços não ocupados do objeto. A variável de decisão x_{de} corresponde ao número de itens de tamanho $e - d$, alocados em um objeto à distância d unidades do início do objeto. A variável $z_k, k = 1,2,\ldots,K$, corresponde à quantidade de objetos de tamanho W_k utilizados. As restrições representadas por (2.14) são restrições de conservação de fluxo que asseguram que o fluxo corresponda a um empacotamento válido, visto que um item pode ser alocado na borda do objeto ou imediatamente após outro item. O conjunto de restrições representado por (2.15) garante que a demanda seja atendida e as restrições representadas em (2.16) limitam a quantidade de objetos utilizados à quantidade disponível para cada classe. Carvalho (2002) prova que esta formulação é correspondente ao problema de balanceamento de máquina de Gilmore e Gomory.

O modelo de *Uns Consecutivos* trata-se da aplicação de uma transformação unimodular no modelo de *Fluxo em Arco.* Esta transformação consiste em somar

cada restrição de conservação de fluxo à restrição de conservação de fluxo anterior, exceto à primeira. Desta forma, a variável x_{de} assume o valor "1", se um item de tamanho $e - d$ é alocado à distancia d da borda do objeto e "0", caso contrário.

O modelo de *Um-Corte* ((2.19) – (2.24)) foi apresentado por Dyckhoff (1981) e estendido posteriormente por Stadtler (1988). Neste modelo cada variável de decisão é tratada como uma única operação de corte, diferente dos modelos de Gilmore e Gomory, onde as variáveis de decisão correspondem a um conjunto de operações de corte. Como destacado por Carvalho (2002), a quantidade de variáveis nestes modelos não cresce exponencialmente como no modelo clássico de Gilmore e Gomory.

No modelo de *Um-Corte*, S representa o conjunto de objetos com tamanho $q \in \{W_1, \ldots, W_k\} \subset \mathbb{N}$ e D o conjunto de itens de tamanho $q \in \{w_1, \ldots, w_m\} \subset \mathbb{N}$, assumindo que $S \cap D = \emptyset$. A variável $y_{p,q}$ indica a quantidade de itens de comprimento p que é dividida em: um item de comprimento q e um item residual $p - q$. Os itens residuais são itens intermediários no processo de corte e não fazem parte necessariamente de uma ordem. Os comprimentos residuais fazem parte do conjunto D. É considerada a mesma função objetivo do modelo de *Fluxo em Arco*, conforme (2.19) a (2.24) a seguir.

$$min \quad \sum_{k=1}^{K} W_k z_k \tag{2.19}$$

$$s.a. \quad z_k + \sum_{p \in D: p+q \in S \cup R} y_{p+q,q} \geq \sum_{p \in D: p<q} y_{q,p}, \forall q \in S = \{W_1, \ldots, W_K\}, \tag{2.20}$$

$$\sum_{p \in S \cup R: p>q} y_{p,q} + \sum_{p \in D: p+q \in S \cup R} y_{p+q,q} \geq \sum_{p \in D: p<q} y_{q,p} + N_q, \quad \forall q \in (D \cup R) \setminus S, \tag{2.21}$$

$$z_k \leq B_k, k = 1, 2, \ldots, K, \tag{2.22}$$

$$y_{p,q} \in \mathbb{Z}_+, \qquad p \in S \cup R, q \in D, q < p, \tag{2.23}$$

$$z_k \in \mathbb{Z}_+, \quad k = 1, 2, \ldots, K \tag{2.24}$$

onde N_q é o valor da demanda de itens de tamanho q. As restrições (2.20) indicam que a quantidade de itens de tamanho q que ainda pode ser dividida, deve ser menor ou igual à quantidade de objetos de tamanho W_k utilizados mais a quantidade de itens de tamanho q obtidos de *um-corte* de objetos padrão e residuais de comprimento $p + q$.

As restrições (2.21) garantem que o número de itens de comprimento q que podem ser ainda divididos ou utilizados para satisfazer a demanda deve ser menor ou igual o número de itens de comprimento q obtidos de um-corte com objetos padrão ou residuais, tanto para corte de um item de tamanho p para uma ordem de tamanho q, como para corte de um item de tamanho $p + q$ para uma ordem de tamanho p. A restrição (2.22) garante que o número de objetos utilizados não seja maior que a disponibilidade.

O modelo de *Um-Corte* de Stadtler (1988) utiliza restrições separadas para dois itens gerados por *um-corte*, além de utilizar variáveis de acoplamento para somar a quantidade de itens de comprimento igual nas duas seções de corte.

O modelo estendido, demonstrado por Carvalho (2000), resolve o modelo de Gilmore e Gomory com a adição de colunas extras. Com este método o autor observou melhor convergência no processo de geração de colunas, além de melhor tempo computacional na resolução da relaxação linear de problemas de grande porte.

3 TRABALHOS RELACIONADOS AOS PROBLEMAS DE CORTE E EMPACOTAMENTO

No contexto do atendimento das metas do programa de crescimento do governo soviético, Kantorovich (1939) propõe a formulação de problemas que podem ser adaptados para resolução de uma vasta gama de problemas de organização do trabalho na indústria. Dentre tais problemas, pode-se citar: a minimização de sobras de corte; distribuição de ordens em máquinas com produtividades diferentes; maximização da produção dado um *mix* de produtos; máxima utilização de recursos; maximização de uso de matéria prima e racionalização de utilização de combustível. Ainda, apresenta um método matemático baseado em multiplicadores para resolução dos modelos propostos. Esta obra é um dos primeiros registros de formulação de problemas na área de Pesquisa Operacional e dela surgiu um dos principais modelos matemáticos para o PCE.

Outro trabalho bastante importante relacionado ao PCE, é mostrado por Eisemann (1957), que trata da minimização da perda de corte de bobinas de papel, decorrente da demanda por diversas larguras de bobinas. No modelo proposto por Eisemann, padrões de corte são apresentados juntamente com a perda associada e um problema de minimização destas perdas é resolvido. Os dois principais inconvenientes deste método são os seguintes: 1) Os padrões de corte devem ser determinados *a priori* e, dependendo do tamanho do problema, podem existir milhares ou até milhões de padrões factíveis possíveis, sendo que a enumeração de todos os padrões de corte ainda pode ser um inconveniente mesmo com os recursos computacionais disponíveis atualmente; 2) Problema é resolvido de forma linear, ou seja, a variável que determina a quantidade de cada padrão de corte utilizado pode resultar em um valor não inteiro, acarretando a necessidade de aplicação de métodos de arredondamentos depois de resolvido o problema.

Para contornar o problema da geração de padrões de corte, Gilmore e Gomory (1961,1963) apresentaram um modelo em que se encontra uma solução inicial factível e novos padrões são criados a partir da resolução de um problema auxiliar, ao invés de selecionados em uma vasta coleção de padrões. Este método não dispensa a necessidade de métodos de arredondamento, entretanto elimina a necessidade de enumeração de grande quantidade de padrões de corte.

A complexidade NP-difícil do PCE é demonstrada em Eisenbrand e Shmonin (2006). Revisão sobre PCEs pode ser encontrada em Haessler e Sweeney (1991), que abordam problemas de corte de estoque de 1 e 2 dimensões, além de Dyckhoff (1990) e Wäscher *et al*. (2007), que abordam classificação e nomenclatura. Carvalho (2002) aborda diversas formulações de Programação Linear (PL) para problemas unidimensionais de corte de estoque e BP.

A relevância do PCE tem estimulado a produção de diversos trabalhos com propostas de algoritmos e hibridizações, na busca por soluções cada vez melhores e mais eficientes. Um método exato, chamado MTP (*Martello-Toth Procedure*), que utiliza uma estratégia de *branching* associado a algoritmos de aproximação, é proposto por Martello e Toth (1990) para resolução de problemas do tipo BP.

Vasko *et al*. (1993) apresentam um exemplo da aplicação prática em uma indústria de laminação de aço, onde um algoritmo do tipo *Branch-and-Bound* (B&B) é utilizado para maximizar a utilização de corte a quente, em detrimento de corte a frio, levando em consideração a não degradação de outros critérios de desempenho.

Chen *et al.* (1996) aplicam uma variação da meta-heurística *Simulated Annealing* (SA) para resolver a formulação de Programação Linear Inteira (PLI) de um problema de corte de estoque unidimensional, utilizando análise estatística para verificar os efeitos dos diversos parâmetros na eficiência e precisão da solução. O método consiste basicamente em três fases: 1) geração de padrões de corte; 2) geração e uma solução inicial factível para o PLI; e 3) aplicação de SA para encontrar a solução do modelo de PLI.

Falkenauer (1996) utiliza um Algoritmo de Grupamento Genético Híbrido (AGGH) nos problemas do tipo BP. Neste trabalho o problema de BP é tratado como um problema de grupamento, que consiste em particionar um conjunto *U* de itens em subconjuntos disjuntos de *U*, com objetivo de minimizar a quantidade de subconjuntos. O Algoritmo de Grupamento Genético (AGG) é uma adaptação do Algoritmo Genético (AG) aos problemas de grupamento, dos quais fazem parte o problema de BP*,* já citado, além do problema de *lay-out* e coloração de grafos. O AGGH surge do "casamento" da técnica de AGG com o *Algoritmo de Aproximação Baseado em Critério de Dominância* proposto por Martello e Toth (1990a).

Scholl *et al*. (1997) apresentam o método BISON para solução exata de problemas do tipo BP. BISON consiste em um método híbrido que combina métodos heurísticos, argumentos de limites, procedimentos de redução e um procedimento de

B&B. Nos casos analisados em Scholl *et al*. (1997), o método BISON obteve desempenho consideravelmente superior ao método MTP de Martello e Toth (1990a).

Um método heurístico construtivo denominado Minimum Bin Slack (MBS), é proposto para resolução de problemas do tipo BP por Gupta e Ho (1999).

Com base no trabalho de Gupta e Ho (1999), Fleszar e Hindi (2002) propõem novas heurísticas variantes do método MBS.

Pileggi *et al*. (2005) abordam o PCE de forma integrada ao problema de sequenciamento dos padrões de corte. O sequenciamento de padrões de corte tem por objetivos evitar atrasos no atendimento à demanda e controlar a quantidade de pilhas abertas, ou seja, pilhas de itens com demanda parcialmente atendida.

Loh *et al.* (2008) apresentam um método heurístico baseado no conceito de *Weight Annealing* (WA) para resolução do problema de *Bin Packing*. WA é uma meta-heurística que compartilha características com as meta-heurísticas *Simulated Annealing* (SA) e *Deterministic Anneling* (DA), e permite que a solução obtida por uma heurística gulosa escape de ótimos locais, através da aplicação de pesos em diferentes partes desta solução. A meta-heurística WA, além de considerar a função objetivo, como na SA, por exemplo, considera também informações de iterações anteriores sobre cada parte do espaço de busca, aplicando perturbações em diferentes partes deste espaço, melhorando a exploração da vizinhança.

Woodcock e Wilson (2010) testam um método híbrido de Busca Tabu (BT) e B&B para problemas de designação, que guarda algumas semelhanças aos PCEs, sendo que tarefas (itens), com determinados tempos (tamanhos), devem ser designados a agentes (objetos), respeitando a restrição de tempo disponível de cada agente (capacidade do objeto), minimizando o custo total de designação (sobras de corte). Neste método, subproblemas de PLI são gerados através da meta-heurística BT e posteriormente resolvidos com auxilio de B&B.

Araujo *et al.* (2010) propõem um método heurístico baseado em algoritmo evolucionário para resolução de um problema de corte de estoque com objetos variados e limitados. Os algoritmos evolucionários são técnicas de otimização estocásticas inspiradas pela teoria de sobrevivência do mais adaptado de Darwin. Consistem em, a partir de uma população (conjunto de soluções) inicial aleatória, promover alterações genéticas em que a sobrevivência dos novos indivíduos gerados está condicionada à sua capacidade de adaptação.

Poldi e Arenales (2010) tratam o PCE multiperíodo, onde é considerada a otimização para vários períodos de planejamento, utilizando um modelo de PL com técnicas de arredondamento que pondera a antecipação ou não de itens.

Cui e Yang (2010) tratam o PCE com objetos de múltiplos tamanhos e reutilização de sobras. São consideradas sobras utilizáveis as sobras de corte maiores que um determinado tamanho, sendo que estas são retornadas ao estoque a um determinado custo. É utilizado um algoritmo que combina um modelo de PL seguido de um método heurístico para tratamento do problema residual, com objetivo de reduzir o custo dos estoques utilizados.

Algoritmos baseados em propriedades de arredondamento são propostos por Jansen e Solis-Oba (2010 e 2011). O algoritmo OPT+1 apresentado em Jansen e Solis-Oba (2011) assume que a propriedade de arredondamento MIURP (*Modified Integer Round-up Property*), onde o valor ótimo do problema de minimização inteira é, no máximo, igual ao menor inteiro maior ou igual ao valor ótimo encontrado pela relaxação linear do modelo de Gilmore e Gomory mais “1”.

Nihat e Agarwal (2012) propõem um método meta-heurístico inspirado em Redes Neurais Artificiais (RNAs) denominado *Augmented Neural Networks* (AugNN) para resolução de problemas de BP.

4 MÉTODOS PARA RESOLUÇÃO DE *BIN PACKING*

Neste capítulo são apresentados alguns métodos para resolução de problemas do tipo BP. Primeiramente são mostrados algoritmos de aproximação. Em seguida é mostrado o método *Minimum Bin Slack* (MBS) de Gupta e Ho (1999), além de suas variações propostas por Fleszar e Hindi (2002). Por fim são apresentados métodos para resolução de BPs baseados em procedimentos de busca local.

4.1 ALGORITMOS DE APROXIMAÇÃO

Os métodos de aproximação para resolução de problemas de BP, apresentados a seguir são métodos bastante simples que encontram soluções de forma bastante eficiente, porém, a qualidade da solução pode não ser satisfatória. Estudo sobre o pior caso de desempenho destes métodos podem ser encontrados em Johnson (1974) e Simchi-Levi (1994).

4.1.1 *Next-Fit*

Next-Fit (NF) é um algoritmo de aproximação bastante simples. Os itens, em ordem *crescente* de índice, são designados ao primeiro objeto aberto, enquanto houver capacidade residual suficiente. Um objeto é considerado aberto quando está disponível para alocação de itens em uma determinada iteração. Quando o tamanho de um item for maior que a capacidade residual do objeto aberto atual, este é fechado, ou seja, torna-se indisponível para alocação de itens. Um novo objeto é então aberto e o item é designado a este novo objeto. A complexidade computacional deste método é O(n) (Martello e Toth, 1990b). Uma variação deste algoritmo é o *Next-Fit-Decreasing* (NFD), onde os itens são primeiramente organizados de forma decrescente para então serem designados e, devido ao sortimento dos itens, a complexidade computacional aumenta para O(n log n) (Martello e Toth, 1990b).

4.1.2 *First-Fit*

First-Fit (FF) é um algoritmo semelhante ao NF, entretanto, enquanto no método NF é aberto apenas um objeto por vez, o método FF mantém abertos os objetos que possuem capacidade residual disponível maior que o menor item não

designado. O item atual é então designado ao objeto de menor índice que esteja aberto e cuja capacidade residual seja maior que o tamanho do item. Se nenhum dos objetos abertos tiver capacidade suficiente, um novo objeto é aberto. O algoritmo FF pode ser utilizado com sortimento de itens em ordem decrescente de tamanho, sendo denominado *First-Fit-Decreasing* (FFD), e a complexidade computacional em ambos os casos é O(n log n) (Martello e Toth, 1990b).

4.1.3 *Best-Fit*

No método *Best-Fit* (BF), os itens são designados, em ordem crescente de índice, de forma que, a cada iteração o item atual seja designado ao objeto aberto com menor capacidade residual disponível, desde que esta capacidade seja suficiente para acomodar o item que está sendo designado. Em caso de empate, a preferência é pelo objeto de menor índice e quando nenhum objeto aberto atende as condições de capacidade, um novo objeto é aberto. O algoritmo BF também possui sua versão com sortimento de itens em ordem decrescente de tamanho (*Best-Fit-Decreasing* - BFD) e, em ambos os casos, a complexidade computacional é O(n log n) (Martello e Toth, 1990b).

4.1.4 *Worst-Fit*

A estratégia *Worst-Fit* (WF) é bastante similar à estratégia BF no que diz respeito aos critérios para seleção dos itens a serem designados, entretanto, ao contrário do método BF, o algoritmo WF procura o objeto aberto com maior capacidade residual possível. Tanto o método WF como sua versão com sortimento de itens em ordem decrescente de tamanho (*Worst-Fit-Decreasing* - WFD) são resolvidos em tempo computacional O(n log n) (Martello e Toth, 1990b).

Em geral, as versões das estratégias NF, FF, BF e WF com sortimento de itens produzem melhores resultados em problemas estáticos. Scholl *et al*. (1997) destacam que as heurísticas FFD e BFD apresentam melhor resultado na análise de pior caso, com razão de desempenho assintótico de pior caso de 1,222, ou seja, o desvio máximo do valor ótimo é de no máximo 22,2% mais objetos, sendo que este desempenho melhora conforme diminuiu o tamanho do maior item do problema em relação ao objeto.

4.2 *MINIMUM BIN SLACK* E VARIAÇÕES

A heurística *Minimum Bin Slack* (MBS), proposta por Gupta e Ho (1999) para resolução de PCEs do tipo BP, é um método orientado ao objeto (*bin-focus*), isto é, para cada objeto é selecionado um conjunto de itens que minimize sua capacidade residual. O algoritmo considera a desiginação de itens em ordem decrescente de tamanho, cujo sortimento é realizado em tempo computacional O(n log n) (Fleszar e Hindi, 2002). Cada preenchimento de objeto é determinado por um procedimento de busca que testa todos os subconjuntos factíveis formados pelos itens ainda não designados e seleciona o subconjunto que resulte a menor capacidade residual possível do objeto. Quando o procedimento de busca encontra um subconjunto de itens que preencha o objeto de forma completa (capacidade residual igual a zero) o processo é interrompido. Como mostram Gupta e Ho (1999), o processo preenchimento de um objeto tem complexidade computacional $O(2^s)$, onde *s* representa a quantidade de itens a serem designados, ou seja, apesar do valor de *s* decair conforme os objetos são preenchidos, o procedimento de busca é executado tantas vezes quanto a quantidade de objetos necessária para resolução do problema. Na figura 4, é apresentado a implementação recursiva para o empacotamento de um objeto, conforme mostrado em Fleszar e Hindi (2002).

Figura 6: Procedimento MBS.

```
rotina MBSOnePackingSearch(q)
início
    para r := q até n' faça
    início
        i := Z'_r;
        se t_i ≤ s(A) então
        início
            A := A U {i}
            MBSOnePackingSearch(r + 1);
            A := A \ {i};
            se s(A*) = 0 então Saia;
        fim;
    fim;
    se s(A) < s(A*) então A* := A;
fim;
```

Fonte: Adaptado de Fleszar e Hindi, 2002.

Sendo:

n = quantidade de itens;

n' = quantidade de itens atualizada (itens não designados);

i = número do item;

t_i = tamanho do item i;

t^{min} = tamanho do menor item;

Z = lista de itens;

r,q = índice da lista de itens Z;

Z'_r = r-ésimo item da lista Z;

A^* = conjunto de itens no empacotamento ótimo do objeto;

A = conjunto de itens;

$s(A)$ = espaço remanescente em um objeto devido a designação dos itens contidos em A.

Gupta e Ho (1999) demonstram que a heurística MBS produz resultados melhores que as heurísticas BFD e FFD, com tempo computacional, em geral, levemente superior, devendo levar-se em consideração que os problemas testados por Gupta e Ho (1999) possuem no máximo 70 itens.

O tempo computacional gasto para resolução de um problema pelo método MBS pode ser bastante prejudicado conforme a estrutura do problema e a quantidade de itens. Dependendo dos tamanhos dos itens, o procedimento de busca pode ser forçado a avaliar todas as combinações factíveis possíveis. Um exemplo é o caso destacado por Fleszar e Hindi (2002), onde os objetos possuem capacidade de valor ímpar e os comprimentos dos itens têm valores pares.

Com o objetivo de reduzir o tempo computacional, Fleszar e Hindi (2002) propõem a inclusão de 3 condições adicionais no algoritmo MBS, de forma a evitar cálculos redundantes não alterando a solução do algoritmo original:

1. No *loop para*, se *r* > *q* e o tamanho dos itens Z'_{r-1} e Z'_r é igual, o processamento do item Z'_r é ignorado, visto que o item Z'_r não irá gerar empacotamento melhor que aqueles gerados com Z'_{r-1};

2. Fazendo T_r a soma dos tamanhos dos itens Z'_r ... Z'_n, se no *loop para*, $s(A) - T_r \geq s(A^*)$, então o procedimento é interrompido, visto que não é possível construir um empacotamento melhor que $s(A^*)$ com os itens remanescentes. Esta restrição é particularmente útil quando o tamanho do objeto é muito grande em relação ao tamanho dos itens;

3. No início do procedimento, se $s(A) < t^{min}$, onde t^{min} é o tamanho do menor item, como nenhum item pode ser adicionado ao empacotamento A, o *loop para* é ignorado.

Fleszar e Hindi (2002), além de melhorar consideravelmente o tempo computacional gasto pelo método MBS pela inclusão de condições adicionais para evitar redundância de busca, apresentam novos métodos para resolução de problemas do tipo BP. O primeiro método é uma pequena variação no algoritmo MBS, chamado pelos autores de MBS'. Outros 3 métodos propostos utilizam o método MBS' como base, são eles: *Perturbation* MBS'; *Relaxed* MBS' e *Sampling* MBS'. Sendo o *Perturbation* MBS' o que apresentou resultados mais consistentes juntamente com o MBS', estes métodos serão apresentados a seguir de forma mais detalhada e seus resultados serão analisados neste trabalho. Também são brevemente apresentados os métodos *Relaxed* MBS' e *Sampling* MBS'.

4.2.1 *Minimum Bin Slack*'

O método MBS' consiste um uma pequena modificação no procediento MBS. Os itens que geram maiores problemas para serem combinados são os de maior comprimento, visto que, quanto mais seu comprimento se aproxima do comprimento do objeto, menores são as opções de combinações possíveis para preencher um objeto de forma completa. Como o método MBS é orientado ao objeto, o procedimento tende a preencher de forma completa todos os objetos, e isto pode fazer com que objetos difíceis de serem combinados, sejam deixados para as últimas iterações. Na prática é possível observar que no método MBS, os últimos objetos são geralmente preenchidos de forma bastante ineficiente, respeitadas as características de cada problema. Nos casos onde os itens têm de comprimento pequeno em relação ao comprimento do objeto, não se observa este tipo de

problema. Ainda, como observado por Fleszar e Hindi (2002), o método MBS' pode produzir resutaldos piores que o algoritmo original, MBS, dependendo das características do problema.

O método MBS' cria os empacotamentos da mesma forma que o MBS. A única diferença entre os métodos é no inicio da criação dos empacotamentos, onde é definido um item semente a partir do qual o empacotamento é construído. O item semente trata-se do primeiro item da lista *Z'*, ou seja, o maior item não designado, que permanece fixo no empacotamento atual. Isto força o algoritmo a designar primeiro os itens grandes, mais problemáticos, na tentativa de criar empacotamentos mais equilibrados. O algoritmo segue o procedimento apresentado anteriormente, na figura 4, tendo como diferença, a fixação do primeiro item da lista Z' ($Z_1{}'$) no empacotamento A. O procedimento *MBSOnePackingSearch(q)* é então inicializado com q = 2, ao invés de q = 1 como no MBS. Vantagem computacional é agregada ao método MBS' devido a dois fatores, como o primeiro item é fixo, menos combinações são avaliadas, sendo o tempo computacional do procedimento *MBSOnePackingSearch(q)* reduzido de O(2^s), no caso do MBS original para O(2^{s-1}) (Fleszar e Hindi, 2002). O segundo fator é devido a sobrar menos espaço disponível no objeto, observado principalmente nos problemas com itens grandes em relação ao comprimento do objeto, diminuindo as combinações possíveis remanescentes e acelerando o processo de preenchimento do objeto.

4.2.2 *Relaxed Minimum Bin Slack*'

O método *Relaxed* MBS' segue a mesma rotina do método MBS', entretanto, uma modificação no critério de parada do procedimento *MBSOnePackingSearch(q)* permite que sejam aceitos empacotamentos não completos, ou seja, com capacidade residual maior que zero. Desta forma, a condição de parada do procedimento de empacotamento é alterada de $s(A^*) = 0$ para $s(A^*) <=$ *resíduo permitido*. Primeiramente, o método é executado com *resíduo permitido* igual a zero, então, o método é executado mais algumas vezes com resíduo permitido crescente, sendo incrementado conforme a fórmula (4.1).

$$v = \lceil 0.5\% * c \rceil, \qquad (4.1)$$

onde *v* representa o resíduo permitido e *c* a capacidade do objeto. A quantidade de iterações é limitada pela condição (4.2), sendo 40 iterações um valor proposto por Fleszar e Hindi (2002) a partir de testes preliminares.

$$\min\left\{40, \frac{c}{v}\right\}. \tag{4.2}$$

4.2.3 *Sampling Minimum Bin Slack*'

O método Sampling MBS' consiste em executar um determinado número de iterações, alterando a ordem dos itens a serem designados. A ordem dos itens é determinada de forma probabilística, sendo que, a probabilidade p_i de um item ser escolhido, é calculada pela fórmula (4.3).

$$p_i = \frac{t_i^2}{\sum_{j \in \mathcal{A}} t_j^2}, \tag{4.3}$$

Da fórmula (4.3) pode-se observar que itens com comprimentos maiores, correspondentes às primeiras posições da lista Z', têm maior probabilidade de serem escolhidos. O critério de parada do algoritmo é atingido quando o limite inferior é alcançado ou quando um determinado número de iterações é executado sem que ocorra melhora da solução.

4.2.4 *Perturbation Minimum Bin Slack*'

Este método é caracterizado pela construção sucessiva de novas soluções, partindo de uma solução inicial, através da perturbação da solução corrente. A perturbação da solução se dá pela construção de um novo empacotamento a partir de um item semente e considera para o novo empacotamento todos os demais itens do problema. A seleção da semente é um fator de grande importância para eficácia do método e deve priorizar items alocados em objetos mal preenchidos, ou seja, que estejam com maiores capacidades residuais na solução atual. A seleção da semente é feita de forma aleatória, sendo que a probabilidade é proporcional à capacidade residual do objeto ao qual o item está designado. Esta probabilidade é calculada conforme a fórmula (4.4), a seguir:

$$p_i = \frac{s(b_i)}{\sum_{j=1}^{n} s(b_j)}, \tag{4.4}$$

A construção de novos empacotamentos a partir de itens que estejam designados a objetos mal preenchidos força a construção de novos empacotamentos mais eficientes. Para priorizar a seleção dos itens alocados em objetos com maiores capacidades residuais, a lista *Z'*, utilizada para criação do novo empacotamento, é iniciada pelo item semente e os demais itens são ordenados de acordo com a ordem decrescente de capacidade residual dos objetos aos quais os itens estão designados. Itens que estejam desiginados a objetos cheios não necessitam ser ordenados, salvando tempo computacional. Seguindo esta ordem, o método evita, apesar de não impedir, que objetos com capacidade residual igual a zero sejam perturbados. Os itens selecionados para o novo objeto são retirados dos objetos aos quais estavam designados e, caso algum destes objetos fique vazio, este é imediatamente eliminado da solução. São considerados dois critérios de parada, o atingimento do limitante inferior e um determinado número de iterações sem sucesso na melhora da solução corrente.

4.3 MÉTODOS BASEADOS EM BUSCA LOCAL

A seguir são apresentados três métodos meta-heurísticos baseados em procedimentos de busca local. O primeiro, *Variable Neighbourhood Search* (VNS), foi prosposto por Mladenovic e Hansen (1997) e posteriormente adaptado por Fleszar e Hindi (2002) para resolução de problemas do tipo BP. O segundo trata-se do método *Weight Annealing* (WA), proposto por Ninio e Schneider (2005) e adaptado para problemas do tipo BP por Loh *et al*. (2008). Por fim, o método *Augmented Neural Network* (AugNN), apresentado por Agarwal (2003) e adaptado para problemas do tipo BP por Kasap e Agarwal (2012).

4.3.1 *Variable Neighbourhood Search*

A meta-heurística VNS consiste em um método de busca local proposto por Mladenovic e Hansen (1997) para problemas de otimização combinatória, posteriomente adapdato por Fleszar e Hindi (2002) para resolução de problemas do tipo BP, sendo ainda combinado ao método *Perturbation* MBS' para utilização como

método híbrido. No método VNS, um procedimento de busca local é aplicado a uma vizinhança que é sistematicamente alterada. A vizinhança é denotada por $N_k(x)$, sendo *N* um conjunto de soluções onde *k* representa a *k-ésima* vizinhança de x. A busca local tem como objetivo promover mudanças locais, para encontrar o melhor resultado de uma determinada vizinhança, que é explorada em distâncias crescentes de forma a evitar soluções localizadas.

Fleszar e Hindi (2002) indicam três procedimentos básicos que são executados no método de busca local: *Shaking*, *Local Search* e *Move*. O processo *Shaking* gera uma solução aleatória x' contida na k-ésima vizinhança de x, então o processo *Local Search* encontra um ótimo local x'' em relação à x'. *Move* avalia a nova solução x'' e a compara com a solução corrente x. Se x'' for melhor que x de acordo com os critérios do algoritmo, x'' passa a ser a solução corrente e o processo é reiniciado com *k = 1*, senão, *k* é incrementado (*k = k + 1*) e uma nova iteração é realizada. Abaixo são descritos como são executados cada um destes procedimentos no contexto dos problemas do tipo BP, conforme proposto por Fleszar e Hindi (2002).

Shaking: A *k-ésima* vizinhança de x é definida como um conjunto de soluções que podem ser obtidas através da execução de *k* movimentos em itens distintos da solução x. Os movimentos podem ser de dois tipos: *Transfers* e *Swaps*. No *Transfer* é considerada a transferência de um determinado item *i* de um objeto, para um dos demais objetos da solução cuja capacidade residual seja suficiente. O movimento de *Swap* considera a troca de um determinado item *i* por qualquer outro item *j* designado a um objeto distinto, respeitando as limitações de capacidade do par de objetos envolvidos no *Swap*. A geração de um movimento é feita a partir da escolha aleatória de um item *i*, seguido pela avaliação de todos os movimentos de *Transfer* e *Swap* possíveis envolvendo *i*. Se existirem movimentos possíveis, um deles é escolhido e executado de forma aleatória, gerando uma solução x'. O item *i* envolvido, além de sua contraparte *j* no caso de *Swap*, é marcado como "movido" para que não seja considerado nas demais iterações do processo de *Shaking* atual. Se não existir movimento possível envolvendo *i*, este item é removido da lista de itens não movidos e um novo item *i* é selecionado de forma aleatória, seguido pela determinação dos movimentos possíveis e seleção aleatória de movimento. Definida uma solução x', está é utilizada como referência para o *Local Search*.

***Local Search**:* O processo de busca local é baseado nos movimentos de *Transfer* e *Swap* e, diferente do processo de *Shaking*, uma função objetivo é considerada. Usualmente em problemas do tipo BP a função é avaliada em termos de quantidade de objetos utilizada, entretanto, como destacado por Fleszar e Hindi (2002), várias configurações de solução podem resultar na mesma quantidade de objetos para um dado problema. Observando que boas soluções implicam objetos cheios ou quase cheios, os autores utilizam a função objetivo (4.5), a seguir.

$$maxf(x) = \sum_{\alpha=1}^{m} (l_\alpha)^2, \tag{4.5}$$

onde *m* representa a quantidade de objetos na solução x e l_α indica a carga do objeto *α*, ou seja, o somatório do comprimento do conjunto de itens designado ao objeto *α* (A_α), dado pela fórmula (4.6).

$$l_\alpha = \sum_{i \in A_\alpha} t_i, \tag{4.6}$$

***Move**:* A cada passo do procedimento, todos os movimentos de *Transfer,* e *Swap* possíveis são avaliados e o movimento que resulte o maior incremento da função objetivo é executado. A variação na função objetivo resultante do movimento de *Transfer*, onde um item *i* é transferido de um objeto *α* para um objeto *β*, é dado pela fórmula (4.7). O movimento de *Swap*, onde um item i de um objeto *α* é transferido para um objeto *β* e um item *j* do objeto *β* é transferido para o objeto *α*, tem a variação da função objetivo, calculada de acordo com (4.8).

$$\Delta f = (l_\alpha + t_i)^2 + (l_\beta - t_i)^2 - (l_\alpha)^2 - (l_\beta)^2, \tag{4.7}$$

$$\Delta f = (l_\alpha + t_i - t_j)^2 + (l_\beta - t_i + t_j)^2 - (l_\alpha)^2 - (l_\beta)^2, \tag{4.8}$$

Como movimentos envolvendo objetos cheios (*l* = *c*) não incrementam a função objetivo, os itens designados a objetos completos não são avaliados e considerados para movimento. A melhor solução encontrada neste processo é denominada x'', que então é comparada à solução corrente x.

O algoritmo VNS proposto por Fleszar e Hindi (2002) para problemas do tipo BP parte de uma solução inicial encontrada pelo método MBS'. O processo de *Shaking* é executado para gerar uma solução aleatória na *k-ésima* vizinhança de x. Esta solução é submetida ao procedimento de busca local, de forma a encontrar uma solução ótima local x''. O ótimo local x'' é comparado com a solução corrente x, primeiramente em termos de quantidade de objetos, sendo que, se a quantidade de objetos na nova solução for menor que a quantidade de objetos em x, a solução x'' é aceita e passa a ser a solução corrente. Se a quantidade de objetos em ambas as soluções for igual, as soluções são comparadas de acordo com o valor da função objetivo (4.5) e se $f(x'') > f(x)$, x'' é aceita como solução corrente. Se uma solução melhor não for encontrada na iteração corrente, *k* é incrementado e o processo reiniciado.

4.3.2 *Weight Annealing*

Weight Annealing (WA) é um método meta-heurístico proposto por Ninio e Schneider (2005), sendo utilizado originalmente na resolução do Problema do Caixeiro Viajante. WA compartilha características com as meta-heurísticas *Simulated Annealing* (SA) e *Deterministic Annealing* (DA) (Loh *et al*., 2008). O método inicia com uma solução obtida por um método heurístico; então, um conjunto de pesos é determinado a partir das características desta solução. Uma nova iteração é realizada utilizando os pesos definidos anteriormente. O conjunto de pesos é sistematicamente alterado em cada iteração e o processo é repetido até que o critério de parada seja atingido. O método WA utiliza um agendamento de resfriamento similar ao utilizado no método SA, de forma que, no início quando a temperatura é alta, as alterações nos pesos são grandes e conforme a temperatura diminui, o valor dos pesos tende a "1". Os pesos criam distorções no espaço de busca, aceitados soluções que deterioram o resultado do problema, de modo a expandir a busca por soluções melhores.

O método WA para resolução de PCEs é proposto por Loh *et al*. (2008), que aplicam o método em problemas do tipo BP, sendo então denominado WABP (*Weight Annealing Bin Packing*). Durante o processo de busca a solução é avaliada conforme a função objetivo (4.9), cujo objetivo é reduzir a quantidade de objetos utilizados na solução através da maximização da carga atribuída a cada objeto.

$$maximizar\ f = \sum_{i=1}^{p} (l_i)^2, \tag{4.9}$$

onde *p* representa a quantidade de objetos na solução corrente e l_i representa a carga do objeto, que é a soma dos comprimentos dos itens designados ao objeto. No procedimento proposto por Loh *et al*. (2008), os pesos são atribuídos a cada objeto de forma a alterar o peso aparente dos itens, sendo calculado de acordo com a função (4.10).

$$w_i^T = (1 + Kr_i)^T, \tag{4.10}$$

onde *K* é um valor constante, *T* é o parâmetro de temperatura e r_i é dado por (4.11).

$$r_i = \frac{C - l_i}{C}, \tag{4.11}$$

sendo *C* a capacidade do objeto. O peso atribuído a cada objeto é proporcional à sua capacidade residual, dessa forma objetos com maiores capacidades residuais, ou seja, não tão bem utilizados, tem pesos maiores e, consequentemente, o tamanho aparente dos itens designados a estes objetos é maior, aumentando a carga aparente destes objetos e sua probabilidade de participar do processo de troca (*swap*).

O processo de *swap* tem por função trocar sistematicamente itens entre objetos, com objetivo de melhorar a função objetivo. A alteração dos comprimentos aparentes, ocasionada pela atribuição de pesos aos objetos, permite que trocas que pioram a função objetivo sejam aceitas, principalmente quando os valores de temperatura são mais altos, expandindo o processo e busca por soluções melhores. Loh *et al*. (2008) propõem 4 estratégias de *swap, Swap* (1,0) e *Swap* (1,1), equivalentes respectivamente às estratégias *Swap* e *Transfer* apresentadas por Fleszar e Hindi (2002) e duas estratégias novas introduzidas por Loh *et al*. (2008), *Swap* (1,2) e *Swap* (2,2). O movimento de troca é aceito quando a variação na função objetivo correspondente for maior ou igual a zero.

- *Swap* (1,0) – Um item i do objeto *α* é movido para um objeto *β*, sendo a variação dada pela função (4.12).

$$\Delta f_{(1,0)} = (l_\alpha - t_{\alpha i})^2 + (l_\beta + t_{\alpha i})^2 - l_\alpha^2 - l_\beta^2, \quad (4.12)$$

onde *t* representa o comprimento do item.

- *Swap* (1,1) – É feita a troca do item *i* do objeto α pelo item *j* do objeto *β*. A variação da função objetivo é calculada conforme (4.13).

$$\Delta f_{(1,1)} = (l_\alpha - t_{\alpha i} + t_{\beta j})^2 + (l_\beta - t_{\beta j} + t_{\alpha i})^2 - l_\alpha^2 - l_\beta^2, \quad (4.13)$$

- *Swap* (1,2) – É feita a troca do item *i* do objeto *α* pelos itens *j* e *k* do objeto *β*. A variação da função objetivo é dada por (4.14).

$$\Delta f_{(1,2)} = (l_\alpha - t_{\alpha i} + t_{\beta j} + t_{\beta k})^2 + (l_\beta - t_{\beta j} - t_{\beta k} + t_{\alpha i})^2 - l_\alpha^2 - l_\beta^2, \quad (4.14)$$

- *Swap* (2,2) – São trocados itens *i* e *j* do objeto *α* pelos itens *k* e *l* do objeto *β*. A variação da função objetivo é dada por (4.15).

$$\Delta f_{(2,2)} = (l_\alpha - t_{\alpha i} - t_{\alpha j} + t_{\beta k} + t_{\beta l})^2 + (l_\beta - t_{\beta k} - t_{\beta l} + t_{\alpha i} + t_{\alpha j})^2 - l_\alpha^2 - l_\beta^2, \quad (4.15)$$

Na figura 5 é mostrada a implementação do método WABP conforme proposto por Loh *et al*. (2008). O método possui os seguintes parâmetros com os seus respectivos valores:

- *T* = Temperatura Inicial: permite maiores distorções no peso aparente nas iterações iniciais, aumentando a probabilidade do algoritmo aceitar trocas que deteriorem o valor da função objetivo de forma a escapar de ótimos locais. A temperatura inicial proposta é igual a “1”;

- *Tred* = Fator de redução: fator que determina a redução da temperatura a cada iteração. O valor proposto por Loh *et al*. (2008) é de “0.95”;

- *K* = Parâmetro de escala: constante que controla o tamanho da distorção no comprimento aparente de cada item. Seu valor proposto é igual a “0.05”;

- *nloop* = Número de iterações: controla a quantidade de iterações executadas pelo algoritmo.

- *LB* = *Lower bound*: Número mínimo possível de objetos necessários para designação de todos os itens, equivalente ao menor inteiro maior ou igual a soma dos comprimentos dos itens, dividido pela capacidade do objeto.

Figura 7: Procedimento WABP.

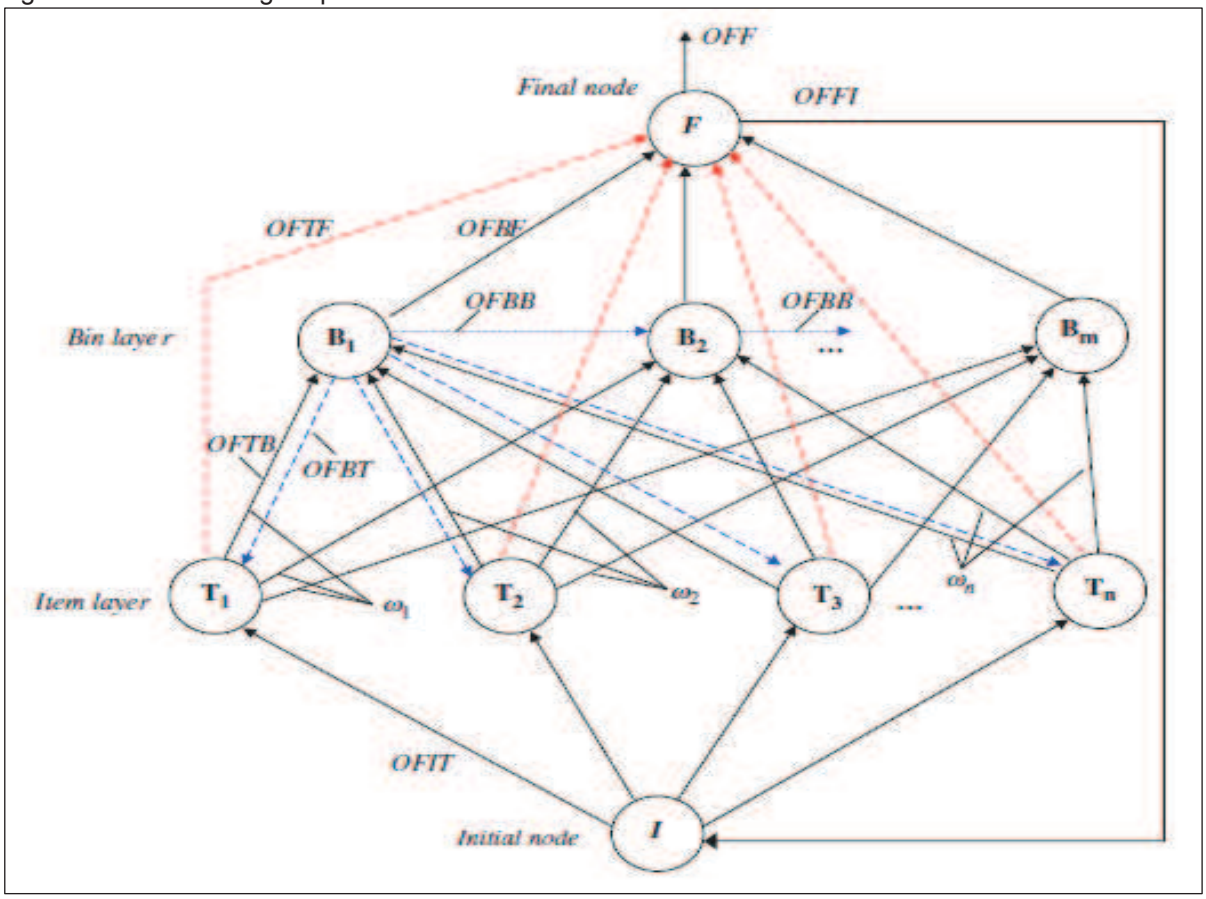
Passo 0. Inicialização.
ajuste os parâmetros
$K = 0{,}05$; *nloop* = 50; $T = 1$; *Tred* = 0,95.
Passo 1. Construção de solução inicial.
resolva o problema utilizando a heurística FFD
compute a capacidade residual dos objetos (r_i)
Passo 2. Melhoramento da solução
para $i = 1$ **até** *nloop* **faça**
compute os pesos $w_i^T = (1 + Kr_i)^T$.
para todos os pares de objetos **faça**
execute *Swap* (1,0)
avalie a factibilidade e $\Delta f_{(1,0)}$.
se $\Delta f_{(1,0)} \geq 0$ **então**
mova o item.
saia do *Swap* (1,0)
se solução = LB **saia** do *loop* i
se movimento factível com $\Delta f_{(1,0)} \geq 0$ não é encontrado **saia** do *Swap* (1,0)
execute *Swap* (1,1)
avalie a factibilidade e $\Delta f_{(1,1)}$.
se $\Delta f_{(1,1)} \geq 0$ **então**
mova o item.
saia do *Swap* (1,1)
se solução = LB **saia** do *loop* i
se movimento factível com $\Delta f_{(1,1)} \geq 0$ não é encontrado **saia** do *Swap* (1,1)
execute *Swap* (1,2)
avalie a factibilidade e $\Delta f_{(1,2)}$.
se $\Delta f_{(1,2)} \geq 0$ **então**
mova o item.
saia do *Swap* (1,2)
se solução = LB **saia** do loop i
se movimento factível com $\Delta f_{(1,2)} \geq 0$ não é encontrado **saia** do *Swap* (1,2)
execute *Swap* (2,2)
avalie a factibilidade e $\Delta f_{(2,2)}$.
se $\Delta f_{(2,2)} \geq 0$ **então**
mova o item.
saia do *Swap* (2,2)
se solução = LB **saia** do *loop* i
se movimento factível com $\Delta f_{(2,2)} \geq 0$ não é encontrado **saia** do *Swap* (2,2)
T:= T x *Tred*
fim do *loop* i
Passo 3. Saída de resultados
apresente a quantidade de objetos utilizada, a designação dos itens e a capacidade residual dos objetos.

Fonte: Adaptado de Loh *et al*., 2008.

4.3.3 *Augmented Neural Network*

Augmented Neural Network (AugNN) é um procedimento híbrido que combina um algoritmo heurístico com a meta-heurística Rede Neural Artificial (RNA), sendo aplicado na resolução de problemas de otimização combinatória. A primeira aplicação deste método foi feita por Agarwal (2003) para resolução de um problema de agendamento de tarefas. Na AugNN, o método heurístico é construído em uma estrutura de RNA, com elementos interligados, onde funções de entrada são transformadas em funções de saída através de funções de ativação. Estas funções são modeladas de forma a capturar as características e restrições dos problemas de acordo com a estrutura da heurística utilizada, permitindo grande flexibilidade na adaptação para diferentes métodos. A mudança controlada de pesos entre os elementos de processamento é responsável pela característica de busca local por soluções melhores. Kasap e Agarwal (2012) propõem a AugNN para resolução de PCEs, fazendo uso de uma estrutura como a mostrada na Figura 6.

Figura 8: Estrutura AugNN para PCEs.



Fonte: Kasap e Agarwal (2012).

A notação utilizada para a AugNN da Figura 6, com seus respectivos significados é a seguinte:

n = número de itens;

m = número de objetos (UB);

T = conjunto de itens (1, 2,...,*n*) (Camada de itens/camada de entrada).

B = conjunto de objetos (1,2,...,*m*) (Camada de objetos/camada escondida).

k = número de iterações.

t = iteração de designação [0,*n*].

I = nó inicial.

F= nó final.

T_i = i-ésimo nó da camada de itens, $i \in T$.

B_j = j-ésimo nó da camada de objetos, $j \in B$.

$OFI(t)$ = função de saída do nó inicial.

$OFTB_i(t)$ = função de saída dos nós dos itens T_i para os nós dos objetos, $i \in T$.

$OFTF_i(t)$ = função de saída dos nós dos itens T_i para o nó final, $i \in T$.

$OFBF_j(t)$ = função de saída dos nós dos objetos B_j para o nó final, $j \in B$.

$OFBT_{ji}(t)$ = função de saída dos nós dos objetos B_j para os nós dos itens $T_i, i \in T, j \in B$.

$OFBB_j(t)$ = função de saída do nó do objeto B_j para o objeto $B_{j+1}, j \in B, j \neq m$.

$OFFI(t)$ = função de saída do nó final.

$OFF(k)$ = função de saída do nó final na iteração k.

$\omega_i(k)$ = peso da ligação entre os nós dos itens T_i para os nós dos objetos na iteração $k, i \in T$.

As funções de entrada, ativação e saída, reproduzem as regras da heurística utilizada. Kasap e Agarwal (2012) aplicam o método AugNN utilizando as heurísticas BFD e FFD e Almeida e Steiner (2013a, 2013b) utilizam FFD como base para a estrutura da AugNN.

Os pesos atribuídos a cada nó da camada de itens são responsáveis pela mudança do comprimento aparente do item, alterando a sequência de designação dos itens aos objetos. Os pesos são modificados a cada iteração, de modo a permitir que a rede faça uma busca por soluções melhores através da alteração na permutação dos itens. A estratégia apresentada em Agarwal (2009) para ajuste de

pesos busca soluções melhores, partindo da solução inicial obtida pelo método heurístico. Para um dado número aleatório *Rnd* ∈ [0, 1], os pesos são alterados de acordo com a função (4.16):

$$\omega_i(k+1) = \begin{Bmatrix} \omega_i(k) + (\alpha lpha * Rnd * \varepsilon * S_i), & se\ Rnd < 0.5, \\ \omega_i(k) - (\alpha lpha * Rnd * \varepsilon * S_i), & caso\ contrário. \end{Bmatrix}, \forall\ i\ \in T, \quad (4.16)$$

onde o erro ε é dado pela diferença entre a solução corrente e o limitante inferior.

Quando há uma melhora no resultado em relação à iteração anterior, um reforço no último conjunto de pesos é aplicado seguindo a função (4.17), a seguir:

$$\omega_i(k) = \omega_i(k) + RF * (\omega_i(k) - \omega_i(k-1))\ \forall\ i\ \in T. \quad (4.17)$$

Para prevenir que a rede siga um caminho que não gere soluções melhores por muitas iterações, é aplicado o mecanismo de *Retorno*, que consiste em, após um número determinado de iterações sem melhoria da solução, reiniciar os pesos com o melhor vetor de pesos encontrado até o momento.

Estratégias alternativas para alteração dos pesos são propostas por Almeida e Steiner (2013b) e *insights* sobre determinação dos parâmetros da AugNN, como *alpha*, *RF* e *número de iterações* podem ser encontrados em Almeida e Steiner (2013a e 2013b).

5 METODOLODIA

Todos os métodos foram codificados em Microsoft Visual Studio 2010® e executados em um computador provido de processador Intel® Core™ I3-2310M de 2,1GHz com memória RAM de 4GB, utilizando sistema operacional Windows 7 Home Premium. Para realização dos testes foram utilizados dois conjuntos de problemas *benchmark* que podem ser encontrados a partir da página *OR-Library* (https://files.nyu.edu/jeb21/public/jeb/info.html). O primeiro conjunto pode ser acessado em (https://files.nyu.edu/jeb21/public/jeb/orlib/files/). Este conjunto de problemas é dividido em duas classes, *u* e *t*. Na classe *u*, o comprimento dos itens é uniformemente distribuído entre 20 e 100 e a capacidade do objeto é fixa em 150. Esta classe é dividida em 4 conjuntos, *u*120 (https://files.nyu.edu/jeb21/public/ jeb/orlib/files/binpack1.txt), *u*250 (https://files.nyu.edu/jeb21/public/jeb/orlib/files/ binpack2.txt), *u*500 (https://files.nyu.edu/jeb21/public/jeb/orlib/files/binpack3.txt) e *u*1000 (https://files.nyu.edu/jeb21/public/jeb/orlib/files/binpack4.txt), cada conjunto contém 20 problemas e a quantidade de itens a serem alocados nos objetos é de 120, 250, 500 e 1.000, respectivamente. A tabela 1 resume as características de cada conjunto de problemas da classe *u* além de apresentar a quantidade total de objetos utilizados na melhor solução conhecida para cada conjunto de problemas.

Tabela 1 – Características dos conjuntos de problemas da classe *u*.

Classe	Conjunto	Quantidade de problemas	Quantidade de itens por problema	Intervalo de tamanho dos itens	Tamanho do objeto	Objetos na melhor solução conhecida
u	**u120**	20	120	[20, 100]	150	983
	u250	20	250	[20, 100]	150	2034
	u500	20	500	[20, 100]	150	4024
	u1000	20	1000	[20, 100]	150	8011
Total		**80**				**15052**

Fonte: Autor, 2014.

Na classe *t*, o comprimento dos itens varia entre 25 a 50 e devem ser designados em objetos de tamanho igual a 100. A classe *t* também é dividida em quatro conjuntos, t60 (https://files.nyu.edu/ jeb21/public/jeb/orlib/files/binpack5.txt), t120 (https://files.nyu.edu/jeb21/public/jeb/ orlib/files/binpack6.txt), t249 (https://files.nyu.edu/jeb21/public/jeb/orlib/files/ binpack7.txt) e t501 (https://files.nyu.edu/jeb21/public/jeb/orlib/files/binpack8.txt), cada conjunto contendo

20 problemas com 60, 120, 249 e 501 itens a serem empacotados, respectivamente. Esta classe foi projetada em forma de *triplets*, ou seja, a cada objeto podem ser designados exatamente 3 itens e o valor ótimo é igual a quantidade total de itens dividido por 3, não existindo folga em nenhum dos objetos na designação ótima, o que implica que a solução ótima seja igual o valor do *Limitante Inferior* (LI), tornando o problema difícil de ser resolvido. O resumo das características dos problemas e o total de objetos utilizados na melhor solução são apresentados na tabela 2.

Tabela 2 – Características dos conjuntos de problemas da classe *t*.

Classe	Conjunto	Quantidade de problemas	Quantidade de itens por problema	Intervalo de tamanho dos itens	Tamanho do objeto	Objetos na melhor solução conhecida
t	**t60**	20	60	[25, 50]	150	400
	t120	20	120	[25, 50]	150	800
	t249	20	249	[25, 50]	150	1660
	t501	20	501	[25, 50]	150	3340
Total		**80**				**6200**

Fonte: Autor, 2014.

O segundo conjunto de dados pode ser acessado em (http://www.wiwi.uni-jena.de/Entscheidung/binpp/) e é dividido em dois conjuntos de dados, *data set 1* e *data set 2*. O *data set 1* (DT1) (http://www.wiwi.uni-jena.de/Entscheidung/binpp/bin1dat.htm) é composto por 720 problemas divididos em 36 classes. As 36 classes são formadas pela combinação dos parâmetros *n*, *c* e w_j, que podem assumir os seguintes valores:

- *n* = 50, 100, 200 e 500;
- *c* = 100, 120 e 150;
- w_j = [1,100], [20,100] e [30,100].

O parâmetro *n* representa a quantidade de itens, *c* indica o tamanho do objeto e w_j o intervalo no qual os comprimentos são uniformemente distribuídos. A tabela 3 apresenta a composição de cada classe, resultante da combinação dos parâmetros *n*, *c* e w_j, além da quantidade total de objetos utilizada na melhor solução conhecida de cada classe de problema.

Tabela 3 – Características das classes de problemas do conjunto DT1.

Conjunto	Quantidade de problemas	Itens por problema (*n*)	Tamanho do objeto (*c*)	Intervalo de tamanho dos itens (w_j)	Objetos na melhor solução conhecida
DT1	20	50	100	[1, 100]	545
	20			[20, 100]	657
	20			[30, 100]	746
	20		120	[1, 100]	451
	20			[20, 100]	537
	20			[30, 100]	625
	20		150	[1, 100]	352
	20			[20, 100]	417
	20			[30, 100]	449
	20	100	100	[1, 100]	1039
	20			[20, 100]	1333
	20			[30, 100]	1476
	20		120	[1, 100]	886
	20			[20, 100]	1051
	20			[30, 100]	1191
	20		150	[1, 100]	688
	20			[20, 100]	830
	20			[30, 100]	888
	20	200	100	[1, 100]	2063
	20			[20, 100]	2588
	20			[30, 100]	2905
	20		120	[1, 100]	1715
	20			[20, 100]	2144
	20			[30, 100]	2349
	20		150	[1, 100]	1359
	20			[20, 100]	1612
	20			[30, 100]	1749
	20	500	100	[1, 100]	5176
	20			[20, 100]	6336
	20			[30, 100]	7237
	20		120	[1, 100]	4205
	20			[20, 100]	5176
	20			[30, 100]	5868
	20		150	[1, 100]	3347
	20			[20, 100]	4017
	20			[30, 100]	4371
Total	**720**				**78378**

Fonte: Autor, 2014.

O conjunto *data set 2* (DT2) (http://www.wiwi.uni-jena.de/Entscheidung /binpp/ bin2dat.htm) é composto por 480 problemas, divididos em 48 classes formadas pela combinação dos parâmetros a seguir:

- *n* = 50, 100, 200 e 500;
- *c* = 1000;
- *avgW* = c/3, c/5, c/7 e c/9;
- *delta* = 20%, 50% e 90%;

onde *n* representa a quantidade de itens e *c* a capacidade do objeto. O parâmetro *avgW* representa o tamanho médio do item, que combinado com *delta*, que significa o desvio máximo no comprimento dos itens em relação à média, forma o intervalo no qual o comprimento dos itens é distribuído de forma aleatória. Na tabela 4 é possível verificar as características de cada classe de problema formada pela combinação dos parâmetros *n*, *c*, *avgW* e *delta*, além da soma da quantidade de objetos utilizados na melhor solução conhecida de cada classe do conjunto DT2.

Tabela 4 – Características das classes de problemas do conjunto DT2.

Conjunto	Itens por problema (*n*)	Tamanho do objeto (*c*)	*avgW*	*delta*	Comprimento dos itens	Quantidade de problemas	Objetos na melhor solução conhecida
DT2	50	1000	333,3	20%	[266, 400]	10	176
				50%	[166, 500]	10	170
				90%	[33, 633]	10	167
			200,0	20%	[160, 240]	10	108
				50%	[100, 300]	10	104
				90%	[20, 380]	10	104
			142,9	20%	[114, 171]	10	77
				50%	[71, 214]	10	77
				90%	[14, 271]	10	74
			111,1	20%	[88, 133]	10	60
				50%	[55, 166]	10	60
				90%	[11, 211]	10	62
	100		333,3	20%	[266, 400]	10	340
				50%	[166, 500]	10	341
				90%	[33, 633]	10	327
			200,0	20%	[160, 240]	10	207
				50%	[100, 300]	10	209
				90%	[20, 380]	10	204
			142,9	20%	[114, 171]	10	148
				50%	[71, 214]	10	146
				90%	[14, 271]	10	143
			111,1	20%	[88, 133]	10	117
				50%	[55, 166]	10	114
				90%	[11, 211]	10	114
	200		333,3	20%	[266, 400]	10	671
				50%	[166, 500]	10	662
				90%	[33, 633]	10	662
			200,0	20%	[160, 240]	10	408
				50%	[100, 300]	10	401
				90%	[20, 380]	10	402
			142,9	20%	[114, 171]	10	288
				50%	[71, 214]	10	288
				90%	[14, 271]	10	284
			111,1	20%	[88, 133]	10	230
				50%	[55, 166]	10	223
				90%	[11, 211]	10	229
	500		333,3	20%	[266, 400]	10	1671
				50%	[166, 500]	10	1662
				90%	[33, 633]	10	1663
			200,0	20%	[160, 240]	10	1012
				50%	[100, 300]	10	1013
				90%	[20, 380]	10	1004
			142,9	20%	[114, 171]	10	710
				50%	[71, 214]	10	712
				90%	[14, 271]	10	722
			111,1	20%	[88, 133]	10	559
				50%	[55, 166]	10	561
				90%	[11, 211]	10	560
Total						**480**	**20246**

Fonte: Autor, 2014.

Os critérios de desempenho avaliados neste trabalho são qualidade da solução e tempo de processamento. O tempo de processamento é medido em milissegundos e a qualidade da solução é avaliada em relação à quantidade de vezes que o método encontra a melhor solução conhecida na literatura, além do erro, caracterizado pela diferença entre a solução obtida pelo método e o valor da melhor solução conhecida do problema. Os resultados são apresentados em tabelas onde a primeira coluna indica o nome do conjunto de problemas e a coluna *Problemas* mostra quantos problemas compõem o conjunto. Cinco colunas apresentam resultados referentes à qualidade de solução, são elas:

- *Sucessos*: Mostra quantas vezes o método encontra a melhor solução conhecida na literatura no conjunto de problemas correspondente;

- *Erro total*: Mostra a diferença entre a quantidade de objetos na melhor solução conhecida na literatura e a quantidade de objetos na solução encontrada pelo método analisado, para cada conjunto de problemas;

- *Erro total %*: Mostra a porcentagem que o erro total representa em relação ao total de objetos utilizados na melhor solução conhecida, calculado pela divisão do *Erro total* pela quantidade de objetos na melhor solução;

- *Erro médio*: É a divisão do *Erro total* pela quantidade de problemas no conjunto correspondente;

- *Erro máximo*: Apresenta a maior diferença, em quantidade de objetos, entre o resultado obtido pelo método e a melhor solução conhecida, em cada conjunto de problemas.

As três colunas seguintes apresentam os resultados referentes a tempo de processamento, sendo:

- *Tempo total*: O tempo total, em milissegundos, gasto pelo método na resolução de todos os problemas de cada conjunto de problemas;

- *Tempo médio*: É a divisão do *Tempo total* pela quantidade de problemas no conjunto correspondente;

- *Tempo máximo*: Representa o problema de cada conjunto que gastou mais tempo para ser resolvido, em milissegundos, dentre os problemas do conjunto correspondente.

O limitante inferior (LI), utilizado como critério de parada para os métodos onde esta condição se aplica, é calculado conforme a fórmula (5.1):

$$LI = \left\lceil \frac{\sum_{i=1}^{n} w_i}{C} \right\rceil, \tag{5.1}$$

onde w_i representa o comprimento do item i e C a capacidade do objeto, sendo n o número de itens que compõem o problema.

6 TESTES COMPUTACIONAIS E ANÁLISE DE RESULTADOS

A Seguir são apresentados e analisados alguns resultados obtidos pelos métodos estudados neste trabalho e suas respectivas variações. O conjunto de todas as tabelas com os resultados obtidos pode ser encontrado no Anexo I. Primeiramente são avaliados os algoritmos de aproximação (NF, NFD, FF, FFD, BF, BFD, WF e WFD), seguido pelos métodos baseados em MBS (MBS, MBS' e *Perturbation* MBS') e pelos métodos baseados em busca local (VNS, WABP e AugNN). Por fim, são apresentados os resultados do método híbrido proposto por Fleszar e Hindi (2002) e de uma nova proposta de hibridização.

6.1 ALGORITMOS DE APROXIMAÇÃO

Os algoritmos de aproximação oferecem soluções razoáveis de forma bastante rápida. Os resultados obtidos pelos métodos NF e NFD serão omitidos devido a sua solução ser igual, no melhor caso, ou pior à solução obtida pelos métodos FF e FFD, respectivamente, além de a vantagem computacional ser pouco relevante para o conjunto de problemas analisados neste trabalho (Scholl *et al*., 1997). Apesar da qualidade de solução dos algoritmos de aproximação ser superada por diversos métodos, a velocidade para obtenção de resultados e a facilidade de implementação computacional podem tornar sua utilização atrativa em algumas aplicações. Estes métodos são bastante utilizados como solução inicial para outros métodos, além de serem utilizados como alternativa para tratamento da parte fracionada de soluções de PCEs resolvidos por modelos de PL.

6.1.1 *First-Fit* e *First-Fit-Decreasing*

Os resultados obtidos pelos métodos FF e FFD são apresentados nas tabelas 1 e 2 do Anexo A, respectivamente. Os resultados mostram superioridade em tempo de obtenção de resposta do método FF em relação ao método FFD. Para a unidade de medida utilizada, o efeito do ordenamento dos itens em ordem decrescente do método FFD começa a ser percebido apenas nos problemas com 500 itens ou mais. Para efeito de comparação, observa-se que a heurística FFD registrou *Tempo máximo* de 8 milissegundos para resolução de um problema do conjunto *u*1000, com 1.000 itens para serem designados, contra 2 milissegundos no método FF. Em

relação a qualidade da solução do conjunto de problemas da classe *u*, o método FFD foi superior, registrando *Erro total* aproximadamente 77% inferior em relação ao método FF. Para os problemas do tipo *t*, o desempenho do método FFD foi pior, com *Erro total* 133,6% maior em relação ao método FF. Nos problemas DT1 e DT2 os problemas são estruturados com os itens previamente ordenados de forma decrescente, portanto é possível observar diferenças nos resultados entre os métodos FF e FFD, exceto pelo tempo computacional.

6.1.2 *Best-Fit* e *Best-Fit-Decreasing*

Os resultados dos métodos BF e BFD podem ser encontrados nas tabelas 3 e 4 do anexo A, nesta ordem. A versão com ordenamento de itens, BFD, produz resultados de *Erro total* 75,5% menor em relação ao método BF, no conjunto de problemas do tipo *u.* Já para os problemas da classe *t*, o resultado piora, apresentando *Erro total* 134,8% maior em relação ao método BF.

6.1.3 *Worst*-Fit e *Worst-Fit-Decreasing*

A tabela 5 do anexo A apresenta os resultados obtidos com a heurística WF e na tabela 6 são compilados os resultados do método WFD. A versão WFD mostrou desempenho superior no conjunto de problemas do tipo *u* com *Erro total* 89,5% menor em relação ao método WF. Nos problemas tipo *t*, o resultado foi melhor para o método WF, com um *Erro* 411 objetos, nos problemas do tipo *t*60, *t*120, *t*249, *t*501, contra um *Erro* de 883 no método WFD.

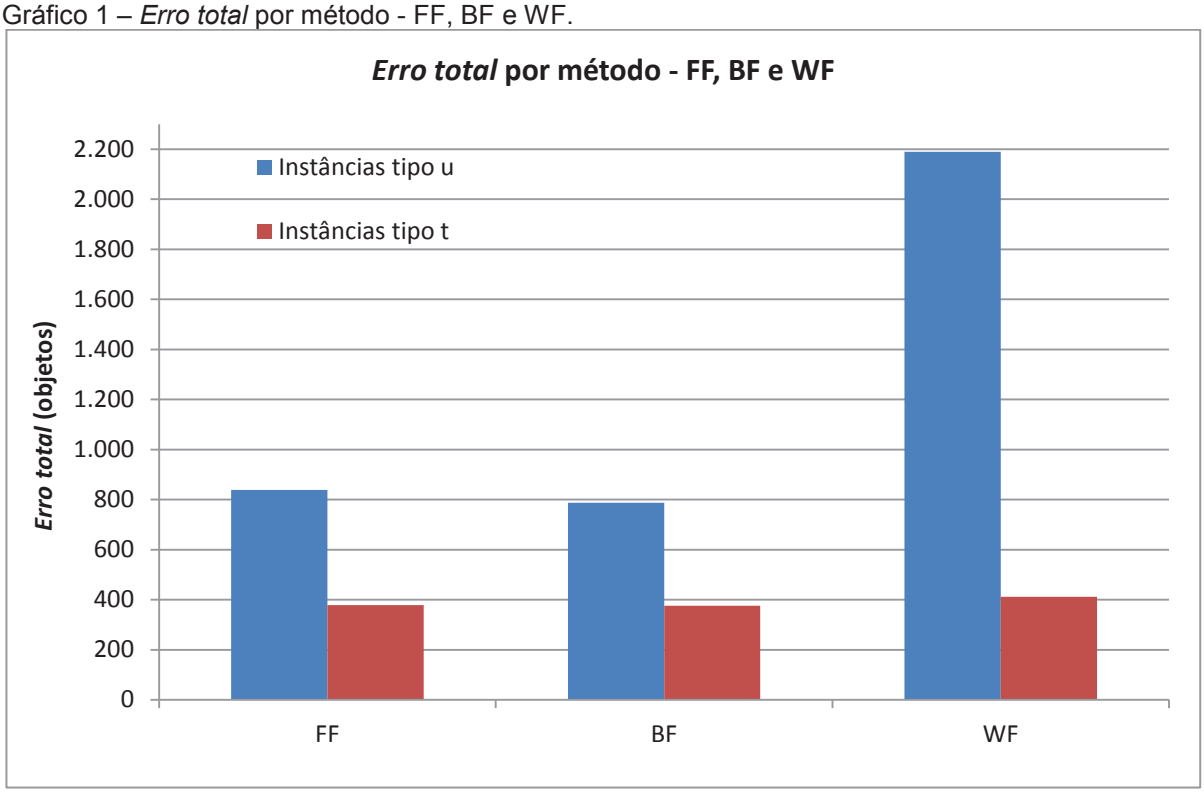
A partir dos resultados obtidos, observa-se que os métodos FF, BF e WF mostraram resultados melhores para os problemas do tipo *t*, donde se conclui que o ordenamento de itens é prejudicial para resolução deste tipo de problema. Nos problemas do tipo *u*, a resolução utilizando os métodos com sortimento dos itens em ordem decrescente, FFD, BFD e WFD, foi mais eficaz em relação a seus pares, FF, BF e WF, respectivamente.

Nos gráficos 1 e 2, a seguir, é possível visualizar a diferença de desempenho entre os algoritmos de aproximação, em termos de *Erro total*, para os métodos sem e com ordenamento de itens, respectivamente.

No gráfico 1 observa-se que nos problemas do tipo *t* os 3 métodos resultaram em *Erros totais* semelhantes, com pequena vantagem para os métodos FF e BF.

Para os problemas do tipo *u*, fica clara a superioridade dos métodos FF e BF, sendo este último levemente superior, com *Erro total* menor em 52 objetos em relação ao método FF.

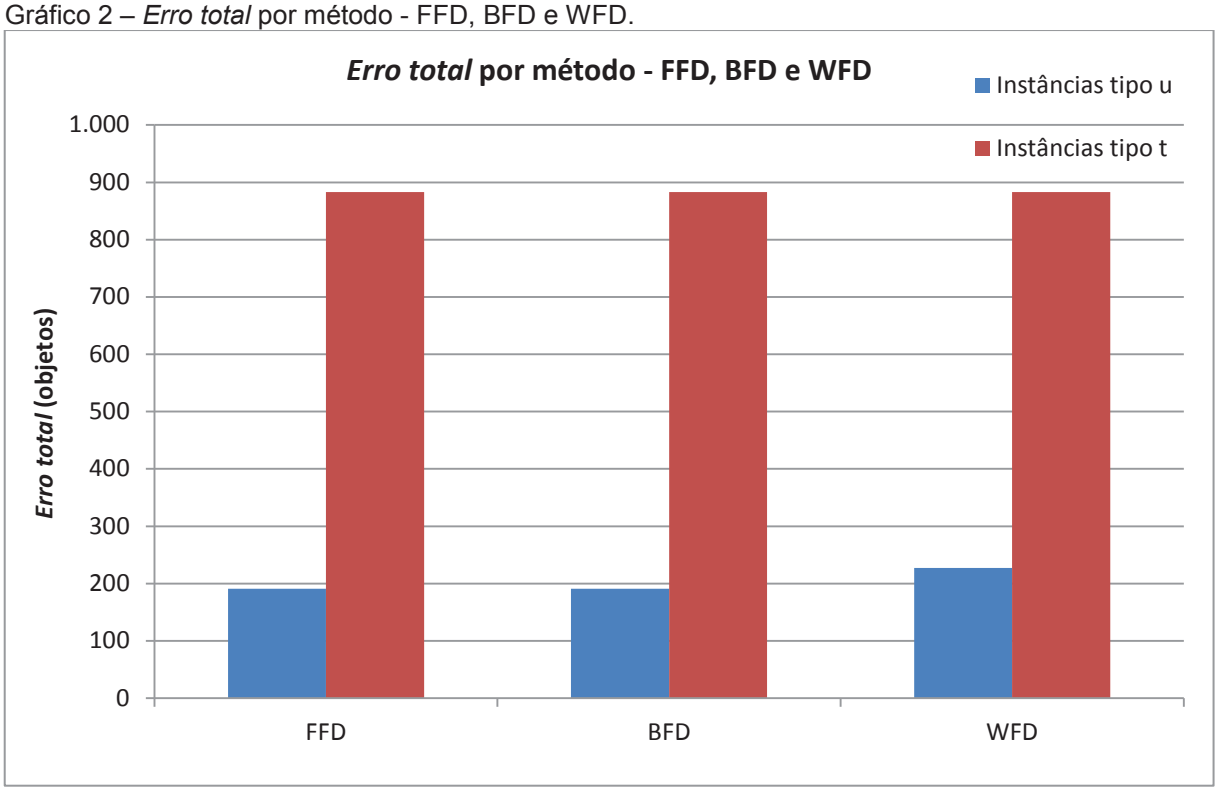
Gráfico 1 – *Erro total* por método - FF, BF e WF.



Fonte: Autor, 2014.

Os resultados obtidos pelos métodos FFD, BFD e WFD, apresentados no gráfico 2, mostram desempenho igual para todos os métodos nos problemas do tipo *t*. Já para problemas do tipo *u*, os métodos BFD e FFD, apresentam resultados levemente melhores que o método WFD.

Gráfico 2 – *Erro total* por método - FFD, BFD e WFD.

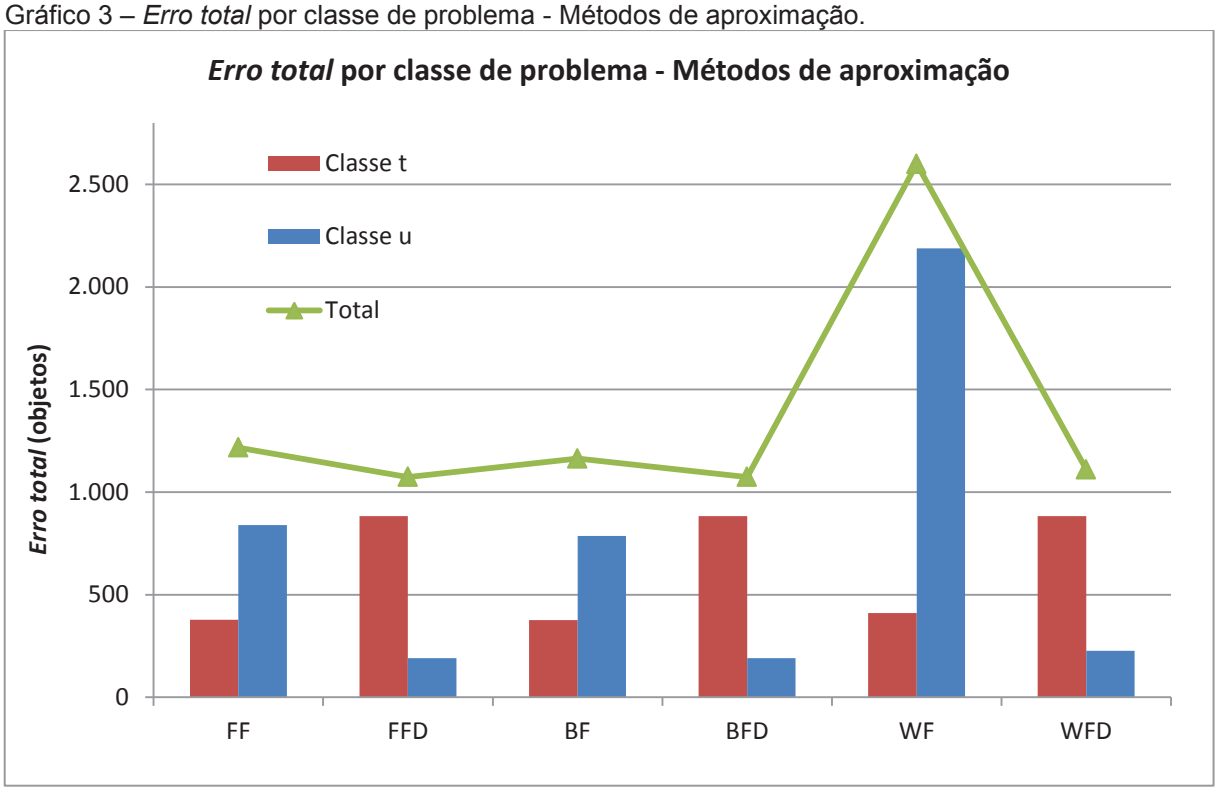


Fonte: Autor, 2014.

No gráfico 3, é possível observar como cada método se comporta para cada tipo de problema. Nos problemas do tipo *u* os métodos FFD, BFD e WFD são mais eficientes enquanto nos problemas do tipo *t* os métodos que produzem os melhores resultados são FF, BF e WF. No geral, somando o *Erro total* nas classes *u* e *t*, os métodos FFD, BFD e WFD foram superiores com *Erro total* de 1.074, 1.074 e 1.110, respectivamente, contra *Erro total* de 1.217, 1.163 e 2.600, nos métodos FF, BF e WF, nesta ordem.

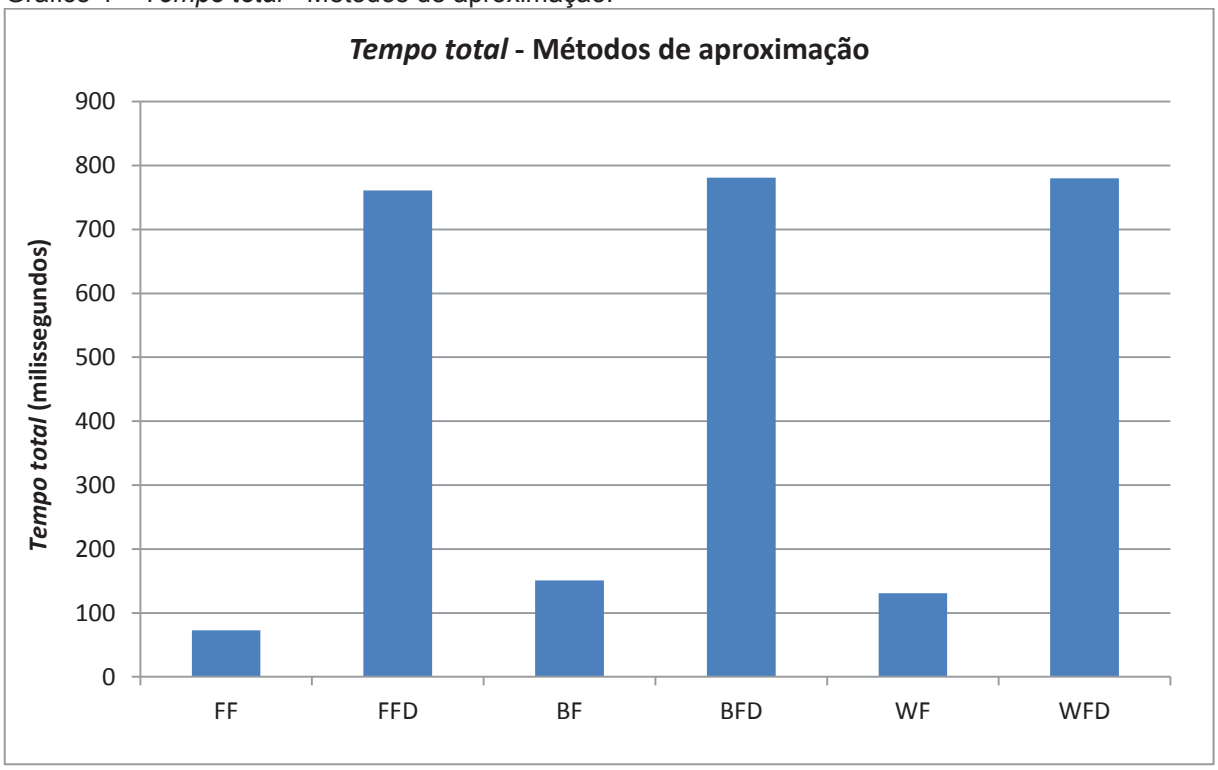
O tempo computacional é mostrado no gráfico 4, onde observa-se um incremento significativo no tempo total para resolução de todos os problemas nos casos onde é aplicado o sortimento de itens em ordem decrescente (FFD, BFD e WFD), entretanto, mesmo com este incremento, o tempo é bastante pequeno, sendo que todos os problemas são resolvidos em tempo inferior a 1 segundo, com média de 0,6 milissegundo por problema.

Gráfico 3 – *Erro total* por classe de problema - Métodos de aproximação.



Fonte: Autor, 2014.

Gráfico 4 – *Tempo total* - Métodos de aproximação.



Fonte: Autor, 2014.

6.2 MÉTODOS BASEADOS EM *MINIMUM BIN SLACK*

A seguir são analisados os métodos MBS de Loh *et al*. (1999), além dos métodos MBS' e *Perturbation* MBS', que são variações do método MBS, propostas por Fleszar e Hindi (2002). O método MBS' produz respostas de forma mais efetiva quando comparado ao MBS, além de servir como solução inicial para outros métodos. Já o método *Perturbation* MBS' foi o único entre os métodos estudados neste trabalho capaz de encontrar respostas ótimas nos problemas do tipo *t.* Os métodos *Relaxed* MBS' e *Sampling* MBS', também propostos por Fleszar e Hindi (2002) não são avaliados devido a seus resultados serem superados, tanto em tempo computacional como em qualidade de solução, pelo método VNS.

6.2.1 *Minimum Bin Slack*

A codificação do algoritmo MBS utilizada neste trabalho considera as condições propostas por Fleszar e Hindi (2002) para eliminar processos redundantes. Testes preliminares mostraram que estas condições podem reduzir o tempo computacional em mais de 50% em relação à implementação original de Loh *et al*. (1999). Os resultados alcançados pelo método MBS são apresentados na tabela 7 no anexo A.

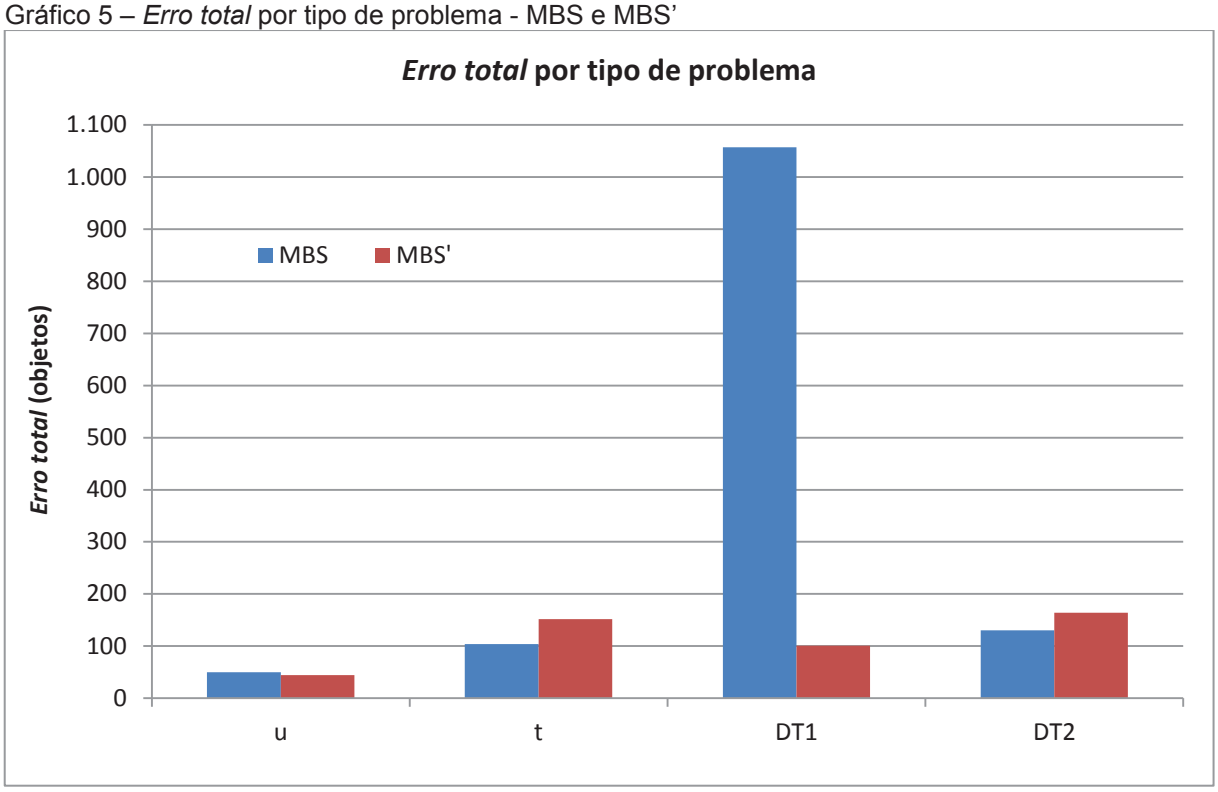
O método MBS necessita de tempo computacional bastante superior em relação aos algoritmos de aproximação para resolução de um problema. Entretanto, nos casos analisadas, os problemas do conjunto u1000, que possuem maior quantidade de itens, foram resolvidos em Tempo médio inferior a 1 segundo. O tempo computacional no algoritmo MBS é bastante influenciado pelas características dos itens a serem empacotados, como em casos observados por Fleszar e Hindi (2002) onde a capacidade do objeto é um valor impar e os itens tem comprimento par, forçando o algoritmo a avaliar todos os empacotamentos possíveis com os itens remanescentes. Nos conjuntos DT1 e DT2 observam-se casos particulares de problemas onde o tempo computacional é bastante superior à média. No conjunto DT1 um problema precisou de 468 milissegundos para ser resolvido, enquanto a média foi de 95 milissegundos. Um caso mais extremo foi observado no conjunto DT2, onde o tempo máximo foi de 15,3 segundos em um problema de 500 itens, contra um Tempo médio de 0,19 segundo do referido conjunto. Ainda, para efeito de comparação, o Tempo médio para resolução de um problema do conjunto u500, que também contém 500 itens, foi de 0,24 segundo.

Em relação à qualidade da solução, o método MBS se mostrou bastante eficiente. Somando os problemas do tipo *u*, o *Erro total* do método MBS foi de 50 objetos, contra *Erro total* de 193 objetos, no melhor caso registrado pelos algoritmos de aproximação (BFD e FFD), ou seja, *Erro total* 74,1% menor. O melhor caso dos métodos de aproximação para os problemas do tipo *t* foi registrado pelo método BF, com soma dos *Erros totais* de 376 objetos, sendo que no método MBS este erro soma 104 objetos, ou seja, melhora de 72,3%. Para o conjunto DT1, o método MBS foi pior que todos os métodos de aproximação, atingindo *Erro total* de 1057 objetos contra *Erro total* de 282 objetos no método BF/BFD, 283 objetos no método FF/FFD e 465 no método WF/WFD. No conjunto DT2 o método MBS foi superior com *Erro total* de 130 objetos contra 748 nos métodos FF/FFD e BF/BFD e 784 no método WF/WFD.

6.2.2 *Minimum Bin Slack*'

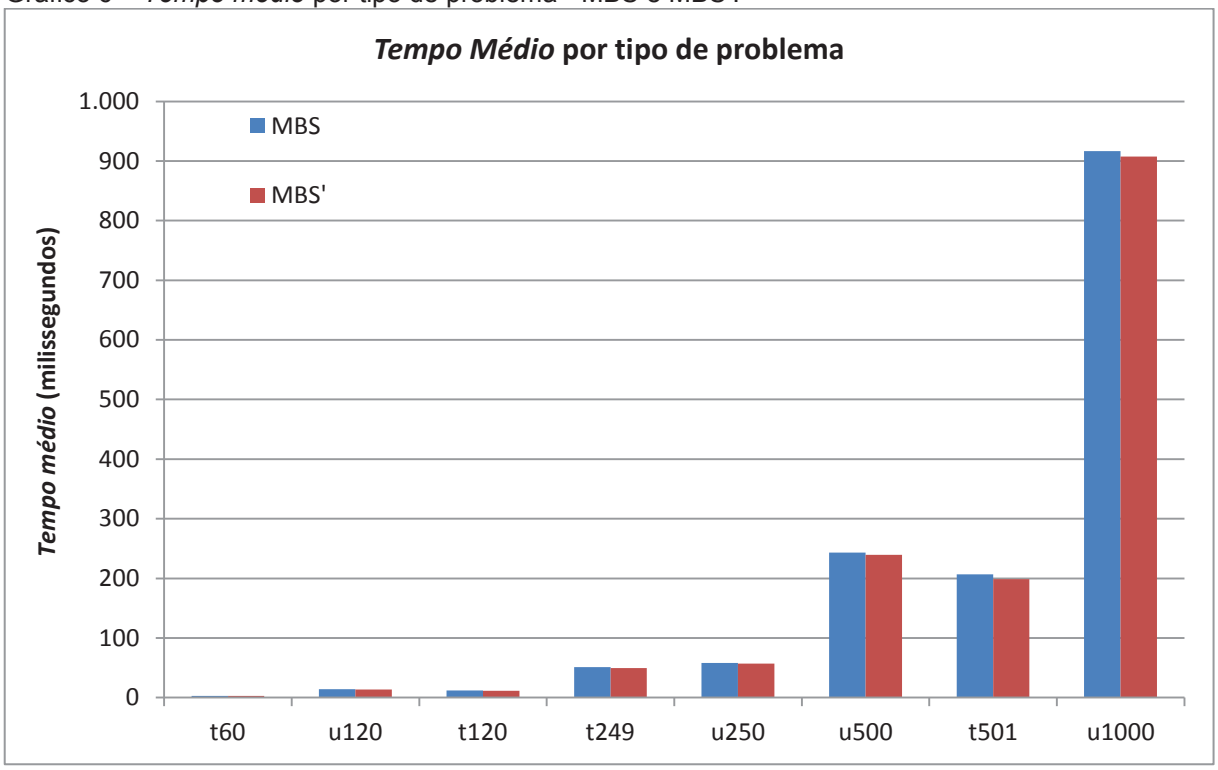
O método MBS' tem como principal característica fixar o primeiro item da lista de itens a serem designados em cada objeto, partindo do pressuposto que os itens maiores são mais difíceis de serem combinados. Uma vantagem evidente deste método em relação ao MBS é a menor quantidade de combinações avaliadas, resultando em menor tempo gasto para resolução de um problema. Os dados da avaliação do método MBS' podem ser encontrados na tabela 8 do anexo A. O *Tempo total* gasto pelo MBS' para resolução de todos os problemas foi 21,9% menor em relação ao método MBS, chegando a uma redução de 39.7% no conjunto DT2. O *Erro total* em todos os problemas utilizando o método MBS' foi de 461 objetos, 65,6% melhor em relação ao método MBS (1341 objetos). Nos problemas do tipo *u* o MBS' foi melhor em relação ao método MBS em 12%, sendo que em um dos problemas do conjunto *u*250 uma solução superior ao melhor resultado conhecido foi encontrada. Nos problemas do tipo *t* a soma do *Erro total* foi 46.1% maior em relação ao método MBS. O método MBS' teve desempenho bastante superior em relação ao MBS nos problemas do conjunto DT1, diminuindo o *Erro total* em 90.4%. No conjunto DT2 o método MBS foi superior ao MBS' com *Erro total* de 130 objetos contra 164 objetos no método MBS'. No gráfico 5 os métodos MBS e MBS' são comparados em termos de *Erro total* e o gráfico 6 mostra a comparação de *Tempo médio*, onde se observa que o tempo para resolução de um problema cresce de forma exponencial, conforme aumenta a quantidade e itens.

Gráfico 5 – *Erro total* por tipo de problema - MBS e MBS'



Fonte: Autor, 2014.

Gráfico 6 – *Tempo médio* por tipo de problema - MBS e MBS'.



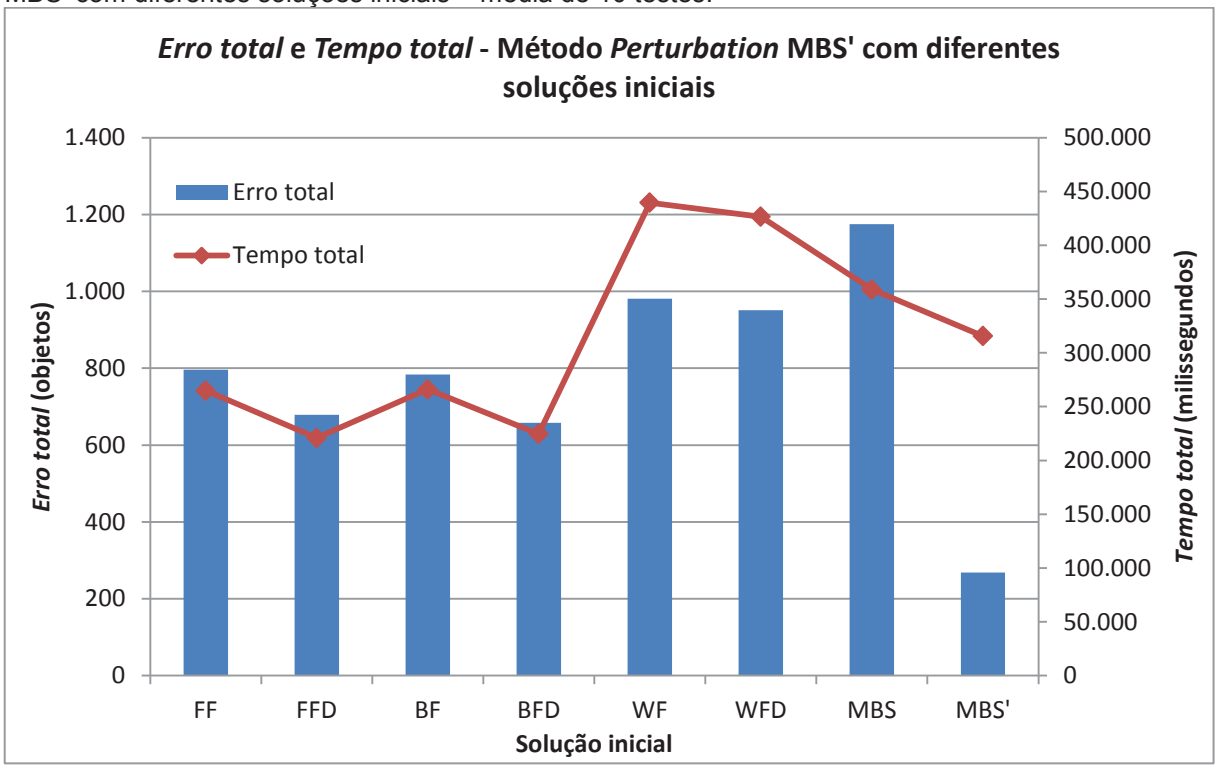
Fonte: Autor, 2014.

6.2.3 *Perturbation* MBS'

Este método parte de uma solução inicial e, por um determinado número de iterações, seleciona um item *semente* o qual serve de base para construção de novos empacotamentos. A escolha da semente é aleatória, sendo que a probabilidade de seleção de um item é maior quanto maior for a capacidade residual do objeto ao qual o item está designado. Neste trabalho, além do método *Perturbation* MBS' proposto por Fleszar e Hindi (2002) que utiliza solução inicial obtida pelo método MBS', sendo aqui denominado de MBS'_*Perturbation* MBS', também foram testadas variações utilizando soluções iniciais obtidas por outros métodos, são eles: FF, FFD, BF, BFD, WF, WFD, MBS, sendo que os métodos resultantes serão denominados FF_*Perturbation* MBS', FFD_*Perturbation* MBS', BF_*Perturbation* MBS', BFD_*Perturbation* MBS', WF_*Perturbation* MBS', WFD_*Perturbation* MBS', MBS_*Perturbation* MBS' e MBS'_*Perturbation* MBS', respectivamente. Os resultados podem ser observados nas tabelas 9 a 16 do anexo A. São utilizados como critérios de parada: a quantidade de iterações sem sucesso, definida em 1.000 iterações; e o atingimento do limitante inferior. Como o método possui um componente aleatório, cada variação foi testada 10 vezes a fim de verificar a robustez do método, sendo que os resultados apresentados referem-se à média dos 10 testes.

O gráfico 7 mostra o *Erro total* das variações do método *Perturbation* MBS', sendo que a variação mais eficaz neste quesito foi a MBS'_*Perturbation* MBS' com *Erro total* médio de 268,3 objetos (*Tempo total* médio = 315,6 segundos), seguido pela variação BFD_*Perturbation* MBS' com *Erro total* médio de 658,2 objetos (*Tempo total* médio = 224,6 segundos) para resolução de todos os problemas. A variação mais eficiente em tempo computacional é a FFD_*Perturbation* MBS', com *Tempo total* médio de 220,9 segundos (*Erro total* médio = 678,6 objetos) para resolução de todos os problemas, seguido pela variação BFD_*Perturbation* MBS' com média de 224,5 segundos (*Erro total* médio = 658,2 objetos). A variação com melhor qualidade de solução, MBS'_*Perturbation* MBS', registrou média de *Tempo total* de 315,6 segundos. A comparação entre o *Tempo total* de todas as variações também pode ser observada no gráfico 7.

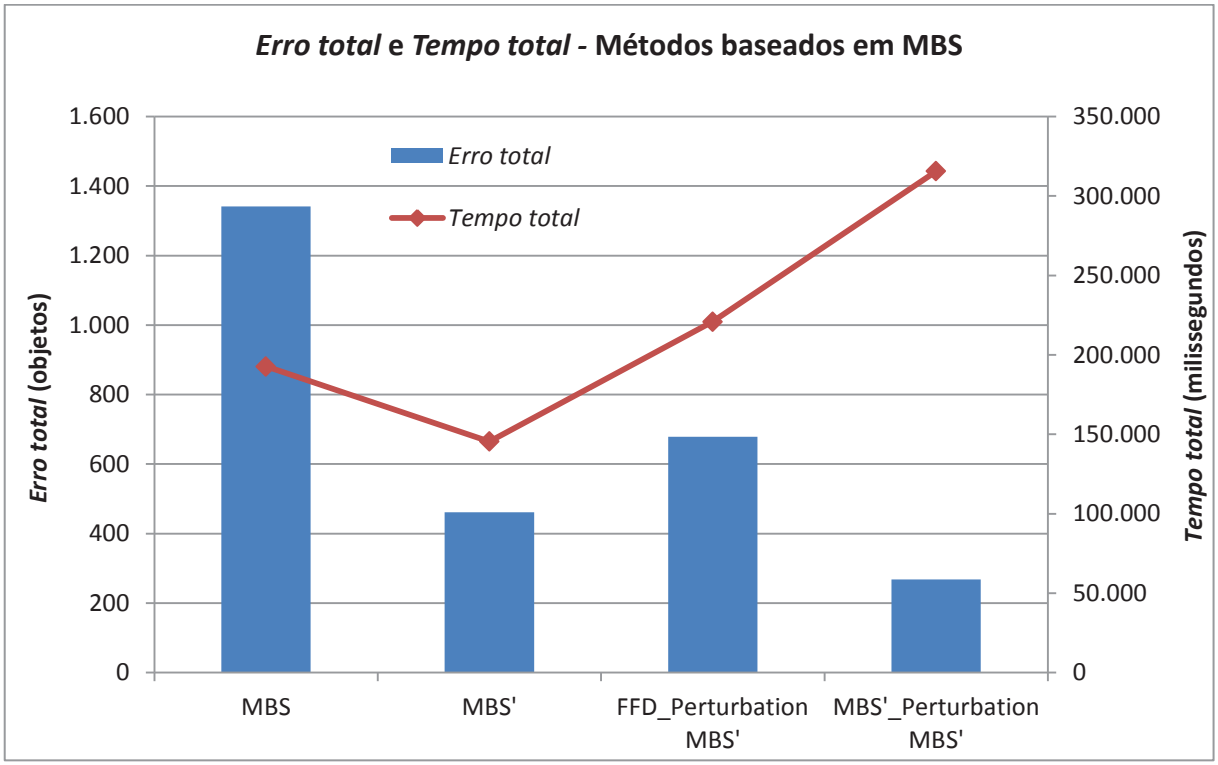
Gráfico 7 – *Erro total* e *Tempo total* para resolução de todos os problemas - método *Perturbation* MBS' com diferentes soluções iniciais – média de 10 testes.



Fonte: Autor, 2014.

Uma comparação entre os métodos MBS, MBS', FFD_*Perturbation* MBS' e MBS'_*Perturbation* MBS' é mostrada no gráfico 8. O método mais eficaz em qualidade de solução é o MBS'_*Perturbation* MBS' com *Erro total* médio de 268,3 objetos (*Tempo total* médio = 315,6 segundos), seguido pelo método MBS' com *Erro total* de 461 objetos (*Tempo total* médio = 145,5 segundos). Em relação ao tempo computacional o método mais eficiente foi MBS' com *Tempo total* de 145,5 segundo para resolução de todos os problemas, seguido pelo método MBS com 192,6 segundos (*Erro total* = 1341 objetos).

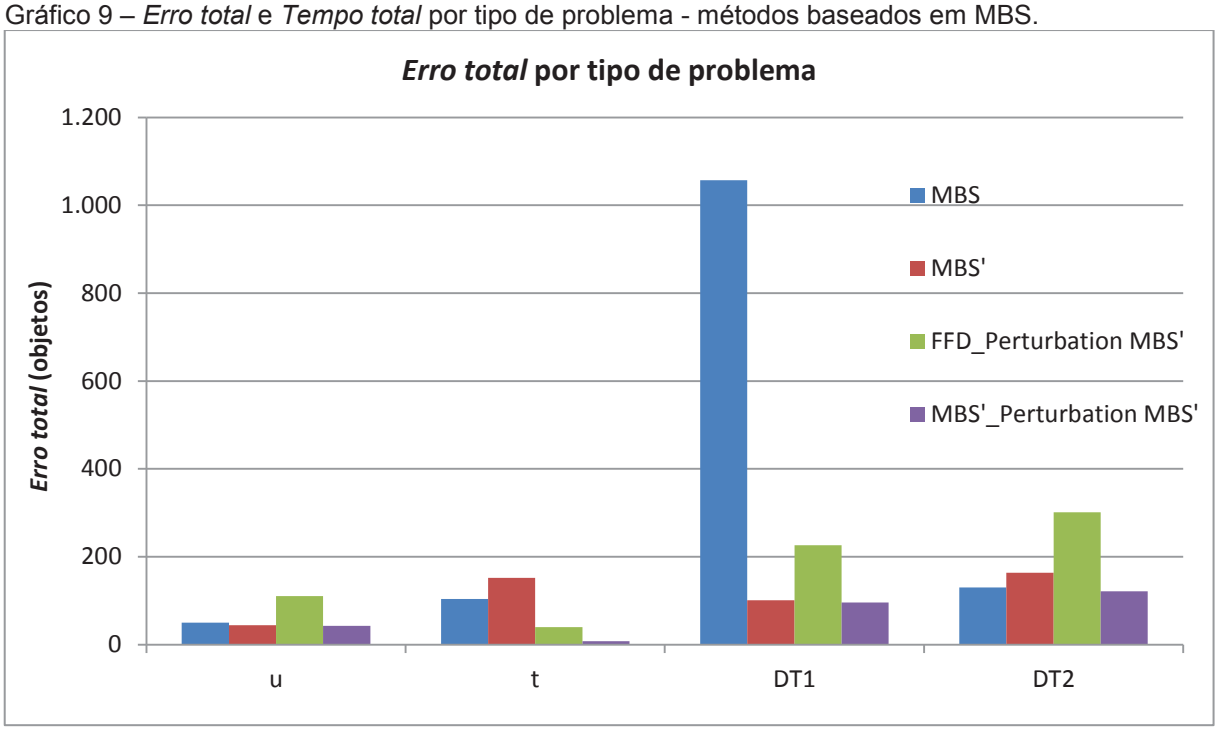
Gráfico 8 – *Erro total* e *Tempo total* para resolução de todos os problemas - métodos baseados em MBS.



Fonte: Autor, 2014.

No gráfico 9, os dados referentes ao *Erro total* apresentados no gráfico 8 são estratificados por tipo de problema. Uma observação importante nestes dados é a eficácia dos métodos FFD_*Perturbation* MBS' e MBS'_*Perturbation* MBS' na resolução dos problemas do tipo *t*, com média de *Erro total* de 39,7 e 7,7 objetos, respectivamente, nos 80 problemas que compõem esta classe. Os métodos que utilizam o procedimento de perturbação da solução foram os únicos, dos métodos avaliados neste trabalho, capazes de encontrar respostas ótimas nesta classe de problemas. O método MBS'_*Perturbation* MBS', além de ser bastante eficiente na classe *t*, superou, mesmo que de forma sutil, os demais métodos baseados em MBS em todas demais classes de problemas, mostrando robustez na resolução de diversos tipos de problemas.

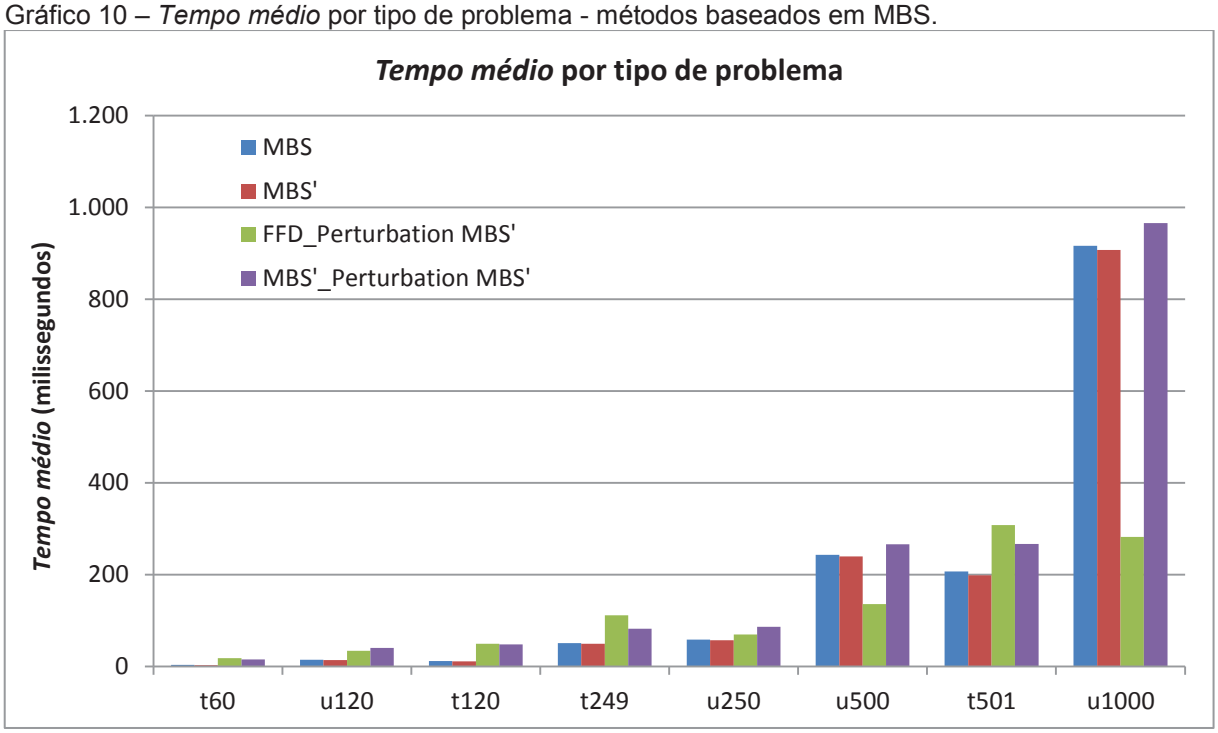
Gráfico 9 – *Erro total* e *Tempo total* por tipo de problema - métodos baseados em MBS.



Fonte: Autor, 2014.

No critério tempo de processamento, é possível observar no gráfico 10 que o *Tempo médio* gasto pelos métodos baseados em MBS aumenta, conforme aumenta a quantidade de itens no problema. Este aumento no tempo acontece, com exceção do método FFD_*Perturbation* MBS', de forma exponencial. Entretanto, mesmo nos problemas maiores estudados neste trabalho, com 1.000 itens, o *Tempo médio* para resolução de um problema é inferior a 1 segundo.

Gráfico 10 – *Tempo médio* por tipo de problema - métodos baseados em MBS.



Fonte: Autor, 2014.

6.3 MÉTODOS BASEADOS EM BUSCA LOCAL

A seguir são apresentados os resultados para os métodos que utilizam estratégias de busca local em seu procedimento de resolução. Os métodos VNS e WA são baseados em um sistema de trocas sucessivas, enquanto o método AugNN utiliza um conjunto de pesos que é sistematicamente modificado de forma a alterar a ordem de desiginação dos itens, de acordo com as regras dos métodos utilizados.

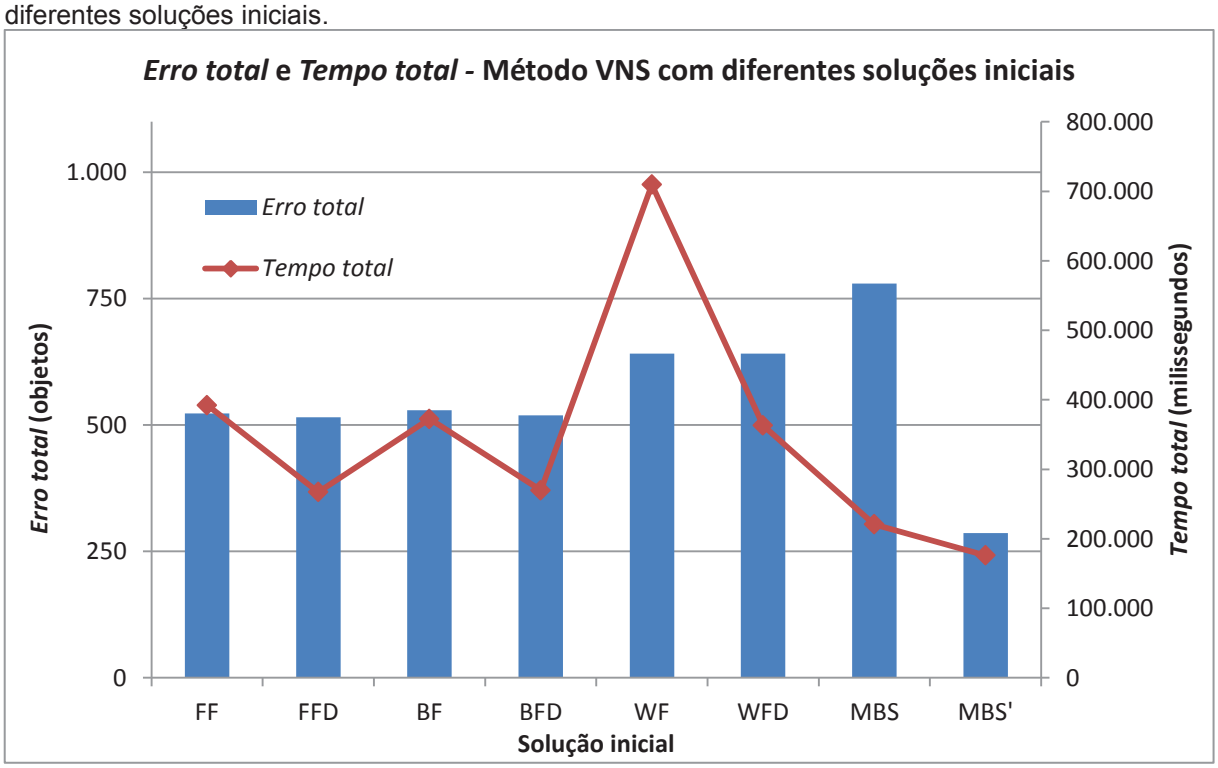
6.3.1 *Variable Neighbourhood Search*

O método VNS utilizado neste trabalho segue a descrição encontrada em Fleszar e Hindi (2002). A quantidade de vizinhanças a serem exploradas, representadas pela quantidade de movimentos em diferentes itens, é definida em *kmax* = 20. Quando um item selecionado para ser movido não possui movimentos possíveis, este pode ser removido temporariamente da lista de candidatos, visto que, após alguns movimentos com outros itens, algum movimento com aquele item pode se tornar viável. Fleszar e Hindi (2002) afirmam que a omissão desta remoção temporária reduz o tempo computacional de O.(kn²) para O.(kn), entretanto pode ocorrer que nenhum dos itens candidatos possua movimentos possíveis. Neste trabalho, a remoção temporária é ignorada e é utilizado um método para evitar *loop*

infinito, quando nenhum dos itens canditados possa ser movimentado, alterando temporariamente a busca de aleatória para determinística. Se nenhum dos candidatos de fato possui movimentos possíveis, o processo é interrompido. Se algum item da lista possuir movimentos possíveis, este é selecionado, o movimento é executado e o processo volta a ser aleatório. Neste trabalho são avaliadas algumas variações do método VNS, sendo utilizando como solução inicial as heurísticas FF, FFD, BF, BFD, WF, WFD, MBS e MBS', sendo denominados FF_VNS, FFD_VNS, BF_VNS, BFD_VNS, WF_VNS, WFD_VNS, MBS_VNS e MBS'_VNS, nesta ordem. Os resultados são apresentados nas tabelas 17 a 24 do anexo A. Como o método VNS possui componente aleatório, o mesmo foi testado 10 vezes para cada solução inicial, sendo os resultados apresentados a média dos 10 testes.

A comparação entre os resultados encontrados com as variações do método VNS pode ser observada no gráfico 11. A variação MBS'_VNS apresentou resultados melhores que as demais, tanto em tempo de processamento como em qualidade de solução, com média de *Tempo total* de 176,1 segundos para resolução de todos os problemas e média de *Erro total* de 286,3 objetos.

Gráfico 11 – *Erro total* e *Tempo total* para resolução de todos os problemas – método VNS com diferentes soluções iniciais.



Fonte: Autor, 2014.

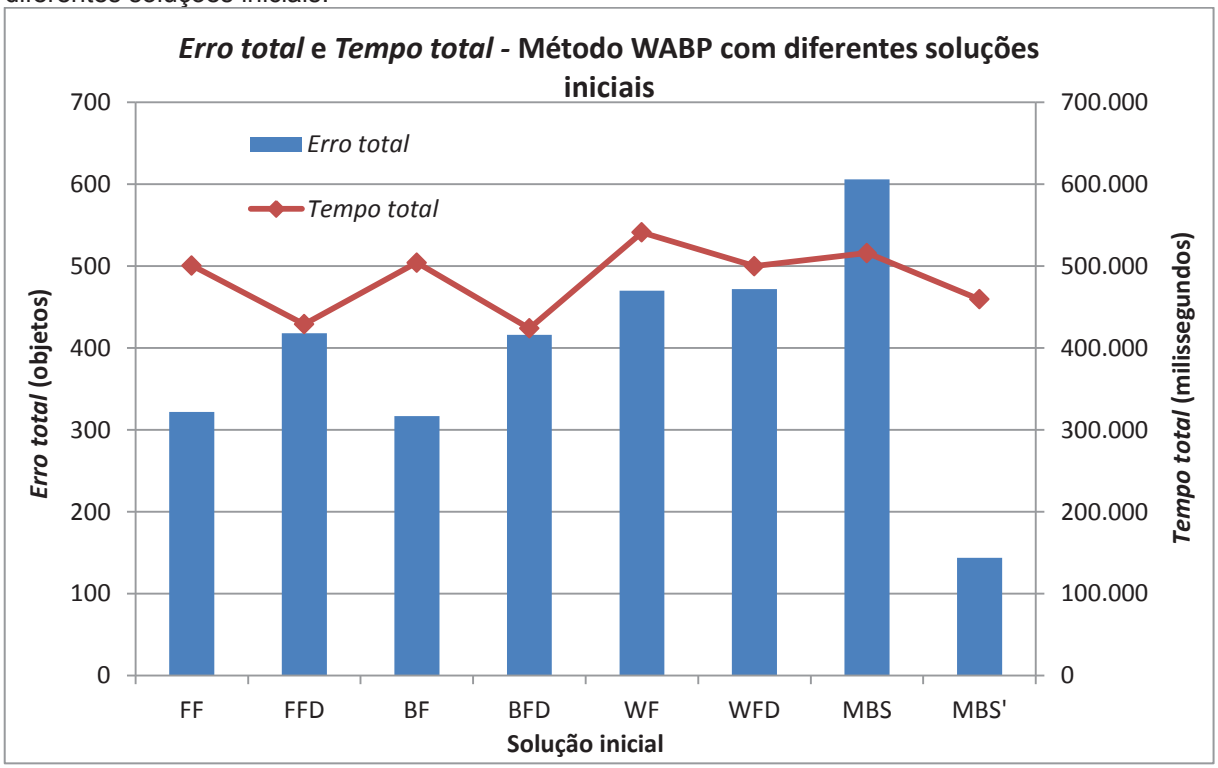
6.3.2 *Weight Annealing Bin Packing*

A meta-heurística WABP apresentada neste trabalho utiliza os mesmos parâmetros apresentados em Loh *et al.* (2008), sendo quantidade de iterações igual a 50, parâmetro de escala K = 0,05, temperatura inicial T = 1 e fator de redução de temperatura *Tred* = 0.95.

Neste trabalho o método foi avaliado utilizando como solução inicial as heurísticas FF, FFD, BF, BFD, WF, WFD, MBS e MBS', sendo denominados FF_WABP, FFD_WABP, BF_WABP, BFD_WABP, WF_WABP, WFD_WABP, MBS_WABP e MBS'_WABP, nesta ordem. Os resultados são apresentados nas tabelas 25 a 32, respectivamente.

O Gráfico 12 mostra o somatório do *Erro total* de todos os problemas e o *Tempo total* para resolução. A variação mais eficaz foi MBS'_WA, com *Erro total* de 144 objetos (*Tempo total* = 459,6 segundos) seguido pela variação BF_WA, com 317 objetos (*Tempo total* = 504,4 segundos). A variação mais eficiente foi BFD_WABP, que resolveu todos os problemas em 424 segundos (*Erro total* = 416 objetos), seguido pela variação FFD_WABP, com 429,1 segundos *(Erro total* = 418 objetos). Em geral, o método WABP é superior em qualidade de solução quando comparado ao método VNS.

Gráfico 12 – *Erro total* e *Tempo total* para resolução de todos os problemas – método WABP com diferentes soluções iniciais.



Fonte: Autor, 2014.

6.3.3 Augmented Neural Network

Neste trabalho, a meta-heurística AugNN é avaliada utilizando como regra de designação as heurísticas FFD, BFD e WFD, definidos com AugNN-FFD, AugNN-BFD e AugNN-WFD, respectivamente. Cada método possui uma regra particular para escolha da sequência em que os itens são designados, na resolução de um problema. No método FFD, escolhe-se o primeiro item da sequência cujo comprimento seja menor ou igual à capacidade residual do objeto. Já o método BFD escolhe o próximo item factível que resulte na menor capacidade residual do objeto que está sendo avaliado, enquanto WFD escolhe o item que resulte na maior capacidade residual. A meta-heurística AugNN, através de um vetor de pesos, altera artificialmente o comprimento dos itens, alterando a sequência na qual os itens são designados, respeitando as regras da heurística utilizada. Como o vetor de pesos influencia no tamanho aparente dos itens e, consequentemente, a ordem na qual são designados, o método AugNN não produz efeito quando aplicado sobre as heurísticas FF, BF e WF pois, nestes casos, o tamanho do item não afeta a ordem em que os itens são avaliados, diferente dos métodos FFD, BFD e WFD, onde os itens são primeiramente classificados em ordem decrescente de comprimento. Os

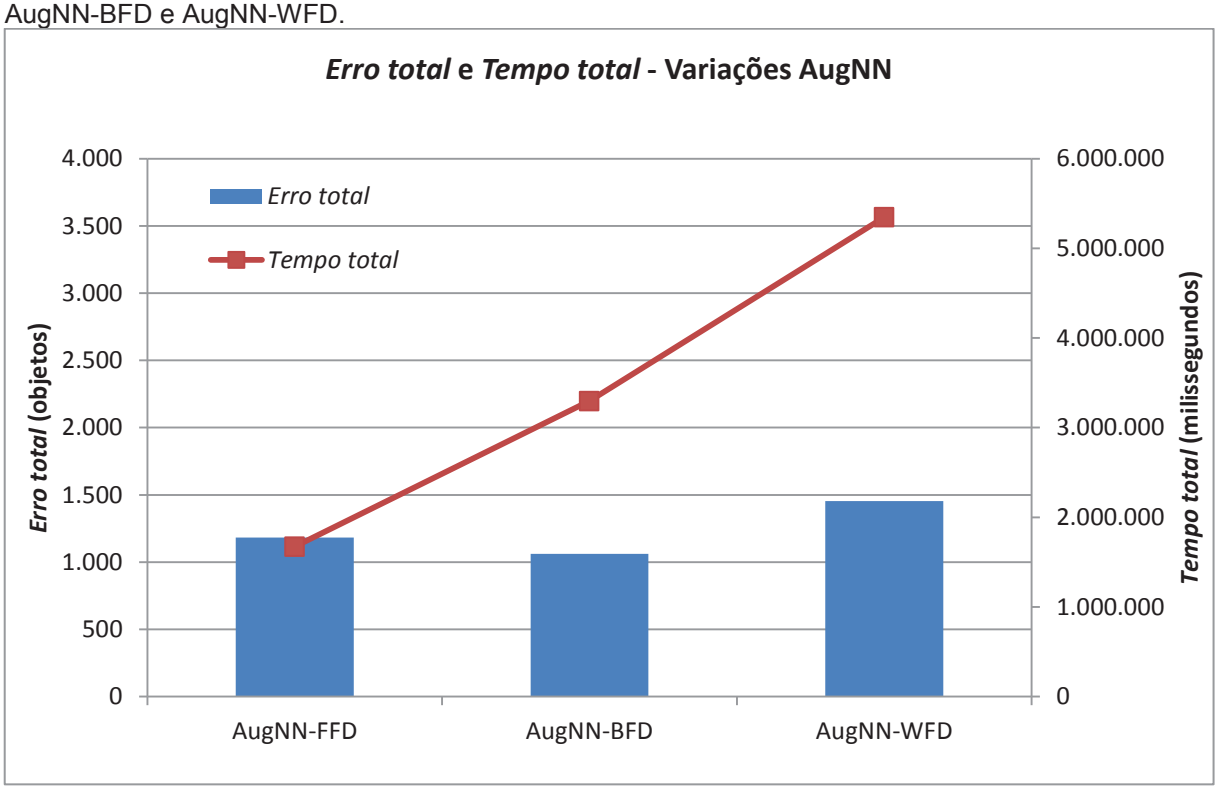
pesos (ω_i) são alterados de forma direcionada utilizando a estratégia proposta em Almeida e Steiner (2013b), descrita na fórmula (6.1):

$$\omega_i(k+1) = \begin{Bmatrix} \omega_i(k) + \left(alpha * Rnd * \varepsilon * S_i * RC_j\right), & \{se\ Rnd < 0.5, \\ \omega_i(k) - \left(alpha * Rnd * \varepsilon * S_i * RC_j\right), & \{caso\ contrário. \end{Bmatrix}, \forall i \in T, \quad (6.1)$$

onde a taxa de aprendizagem ($alpha$) utilizada é de 0,00000353; o número de iterações k definido em 1.500. Rnd representa um valor aleatório tal que {Rnd ∈ [0, 1]}, $\mathrm{S_i}$ é o tamanho do item i e $\mathrm{RC_j}$ representa a capacidade residual do objeto j. ε representa o erro da solução e é calculado pela diferença entre a solução obtida e o limitante inferior. O fator de retorno foi definido em 250 iterações. Se em 250 iterações não houver melhora na solução, os pesos são reiniciados, assumindo o valor do melhor vetor de pesos encontrado até o momento. Cada configuração foi testada 10 vezes para verificar a robustez do método e as tabelas 33 a 35 apresentam os resultados médios das variações AugNN-FFD, AugNN-BFD e AugNN-WFD, nesta ordem.

A meta-heurística AugNN foi capaz de melhorar os resultados dos métodos sob os quais são construídos em todos os casos, diminuindo o *Erro total* em 43,9%, 49,6% e 38,4%, para os métodos AugNN-FFD, AugNN-BFD e AugNN-WFD, em relação às heurísticas FFD, BFD e WFD, respectivamente. Apesar da melhora significativa, este método não apresenta boas soluções, sendo que, no melhor caso (AugNN-BFD), o *Erro total* é superior a 1.000 itens com elevado tempo computacional para resolução, como pode ser observado no gráfico 13, sendo facilmente superado por outros métodos.

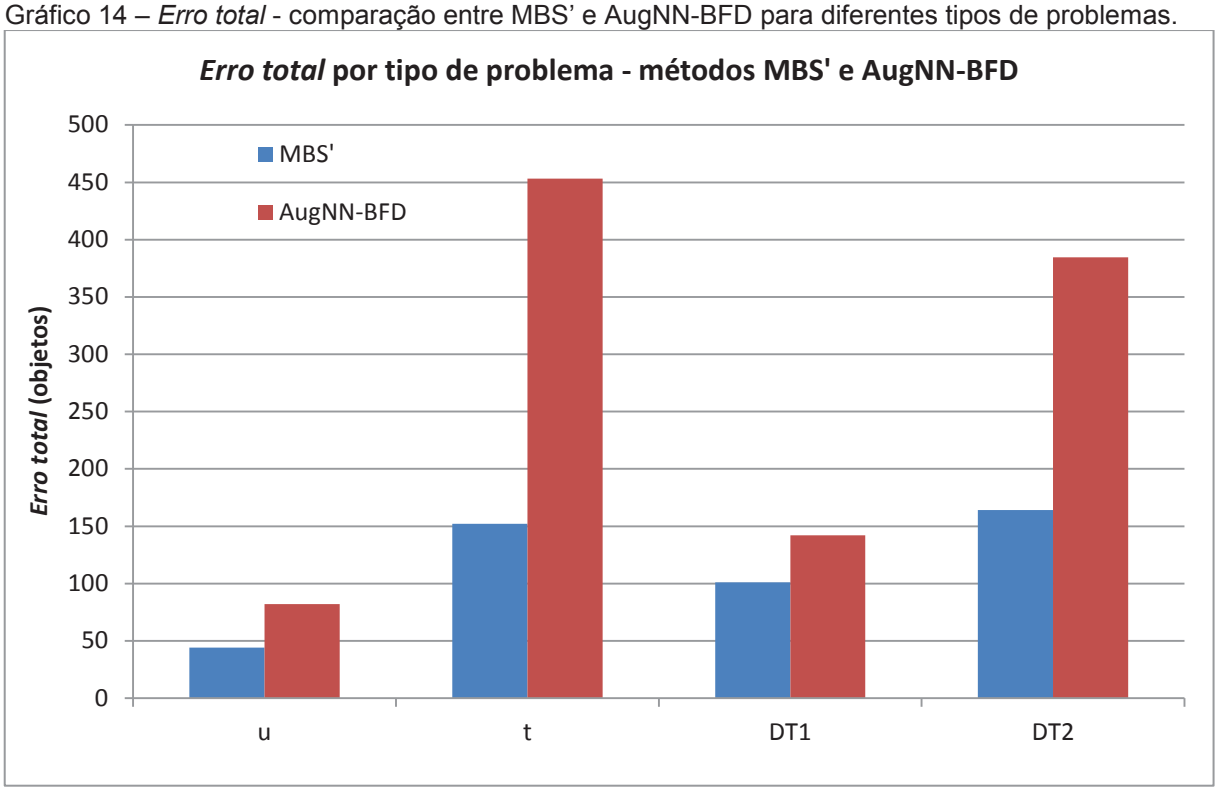
Gráfico 13 – *Erro total* e *Tempo total* para resolução de todos os problemas – métodos AugNN-FFD, AugNN-BFD e AugNN-WFD.



Fonte: Autor, 2014.

Os gráficos 14 e 15 a seguir, mostram a ineficiência e pouca eficácia do método AuNN para resolução dos problemas estudados neste trabalho. Comparando a configuração que obteve melhor qualidade de solução (AugNN-BFD) com o método MBS', por exemplo, o método MBS' foi superior em qualidade de resposta em todos os tipos de problemas (Gráfico 14).

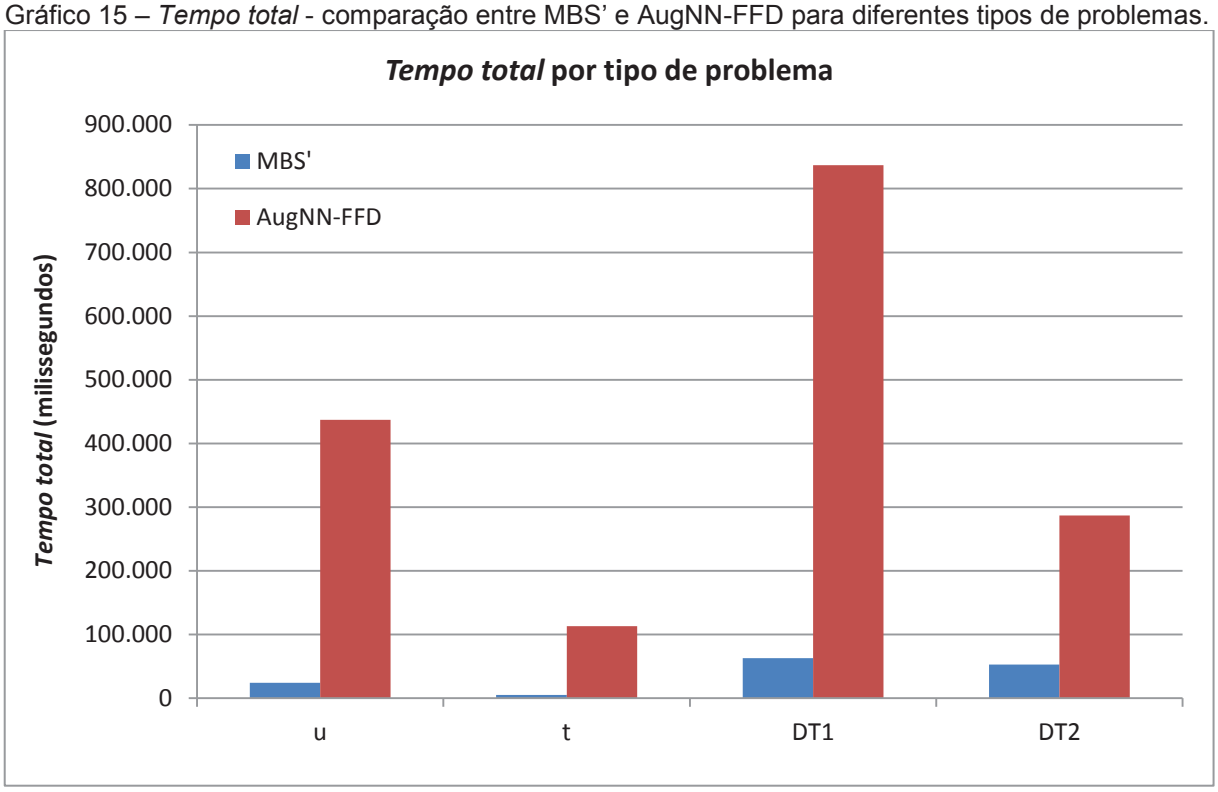
Gráfico 14 – *Erro total* - comparação entre MBS' e AugNN-BFD para diferentes tipos de problemas.



Fonte: Autor, 2014.

Em relação ao tempo computacional a comparação entre a versão mais eficiente (AugNN-FFD) e o método MBS' mostra que este último é bastante superior, como pode ser observado no gráfico 15, ou seja, nos problemas avaliados neste trabalho, o método AugNN é superado em todos os critérios pelo método MBS', o que não justifica sua utilização.

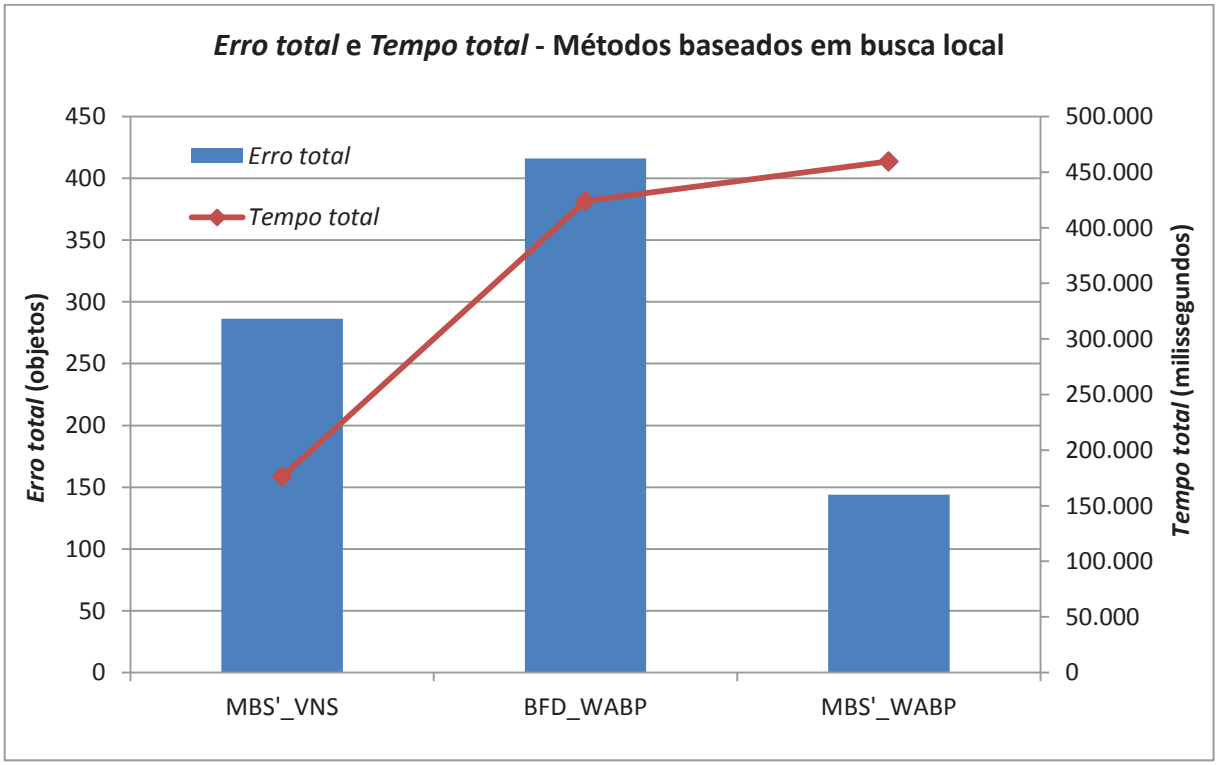
Gráfico 15 – *Tempo total* - comparação entre MBS' e AugNN-FFD para diferentes tipos de problemas.



Fonte: Autor, 2014.

No gráfico 16, a seguir, é mostrada a comparação entre os métodos baseados em busca local MBS'_VNS, variação mais eficiente e eficaz do método VNS e duas variações do método WABP: BFD_WABP (mais eficiente) e MBS'_WABP (mais eficaz). Como as variações do método AugNN são superadas em todos os critérios e em todos os tipos de problema pelo método MBS', estas não serão utilizadas na comparação. O método MBS'_WABP apresentou o menor Erro total, com 144 objetos, 49,7% inferior em relação ao MBS'_VNS.

Gráfico 16 – *Erro total* e *Tempo total* para resolução de todos os problemas - métodos baseados em busca local.

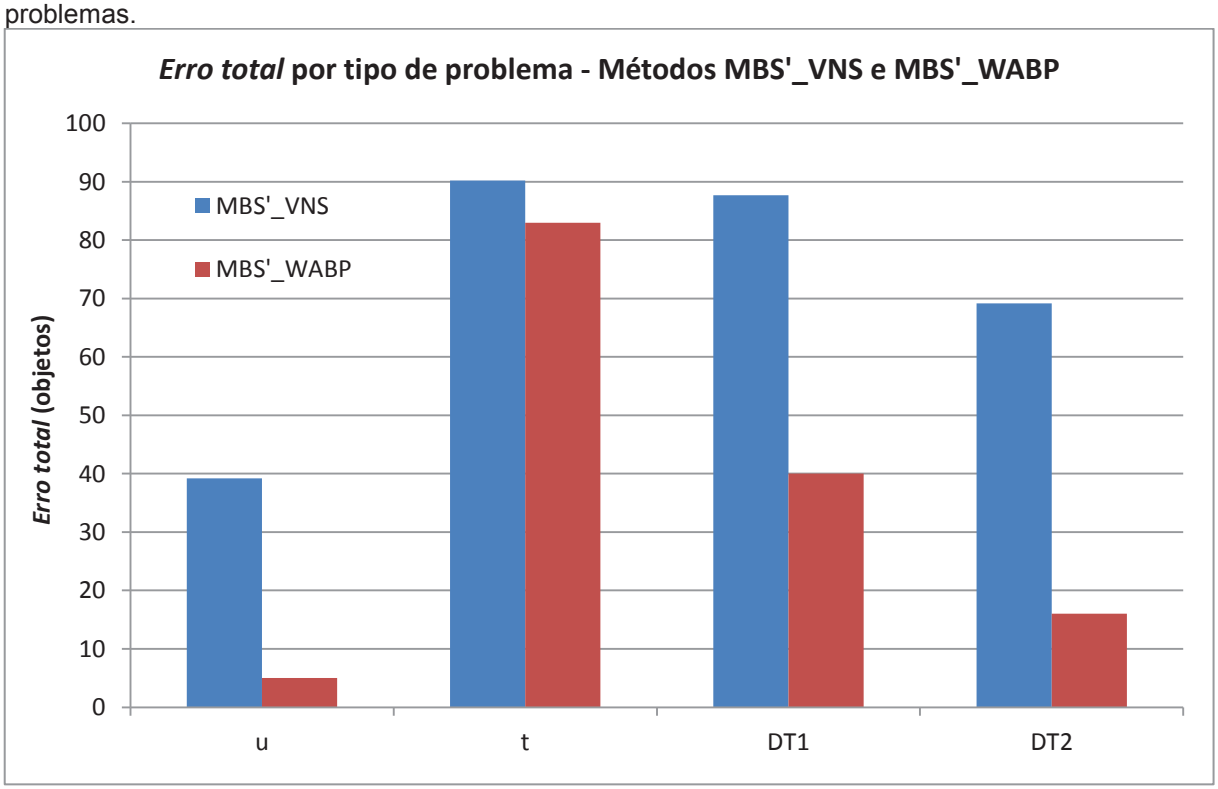


Fonte: Autor, 2014.

O ganho em eficiência registrado pelo método BFD_WABP em relação ao MBS'_WABP é de apenas 35 segundos em 1360 problemas (ganho de 7,7%) no *Tempo total*, enquanto o *Erro total* diminui de 416 objetos no BFD_WABP para 144 objetos no MBS'_WABP, ou seja, uma redução de 65,3%. Ainda, o método MBS'_VNS é superior ao BFD_WABP tanto em tempo de processamento como em qualidade de solução.

No gráfico 17 a seguir mostra a comparação entre os métodos MBS'_VNS e MBS'_WABP por tipo de problema. Observa-se que o método MBS'_WABP é significativamente superior ao método MBS'_VNS, exceto nos conjunto tipo t, em que ambos os métodos não foram capazes de encontrar nenhuma resposta ótima (ver tabelas 24 e 32, Anexo A).

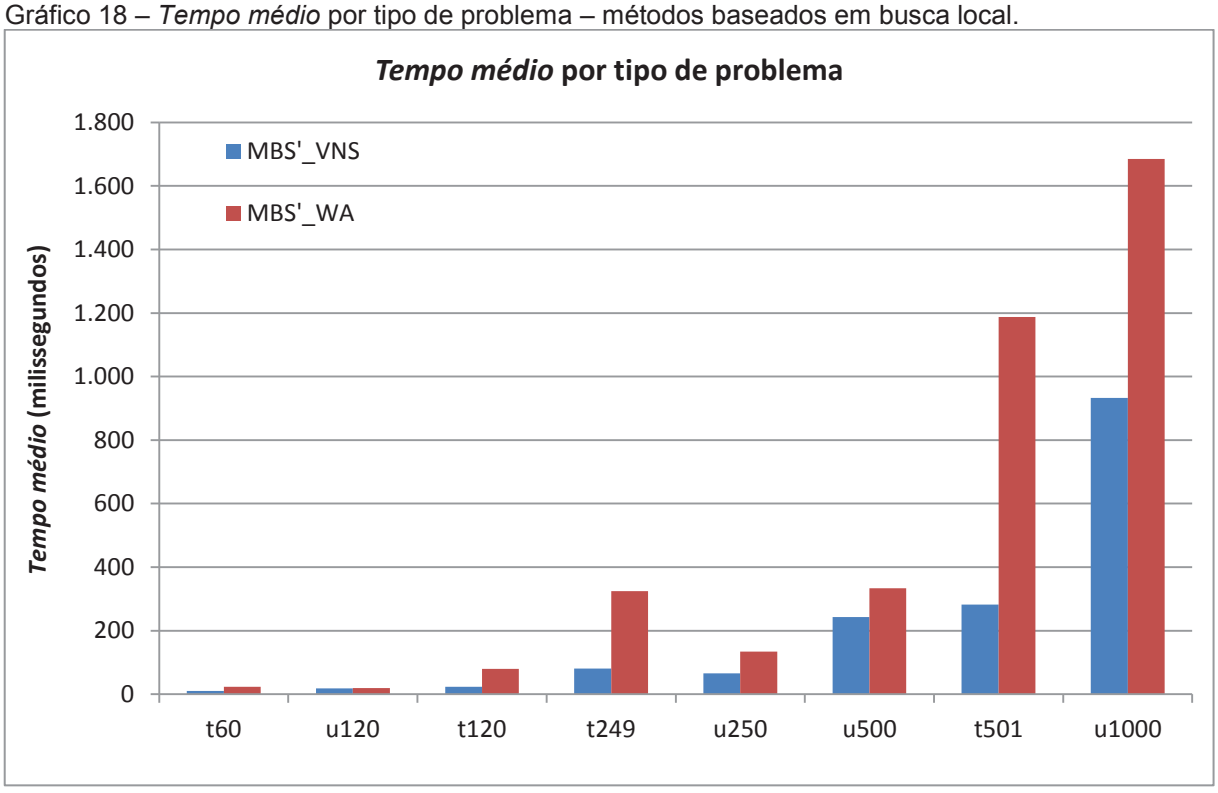
Gráfico 17 – *Erro total* - comparação entre MBS'_VNS e MBS'_WABP para diferentes tipos de problemas.



Fonte: Autor, 2014.

No gráfico 18 observa-se que as diferenças de *Tempo médio* mais significativas acontecem nos problemas do tipo *t*.

Gráfico 18 – *Tempo médio* por tipo de problema – métodos baseados em busca local.



Fonte: Autor, 2014.

As análises e conclusões obtidas pela avaliação dos métodos estudados neste trabalho e suas variações, criam condições para estruturar o método hibrido apresentado neste trabalho. A proposta deste método híbrido é aliar a eficácia do método MBS'_WABP na resolução dos problemas do tipo *u*, DT1 e DT2 à eficácia do método MBS'_*Perturbation* MBS' na resolução dos problemas do tipo *t*.

6.4 MÉTODOS HIBRIDOS

Nesta seção são mostrados os resultados do método hibrido *Perturbation* MBS'+VNS apresentado por Fleszar e Hindi (2002), além de uma nova proposta de hibridização com base nos resultados e análises dos diversos métodos avaliados anteriormente neste trabalho.

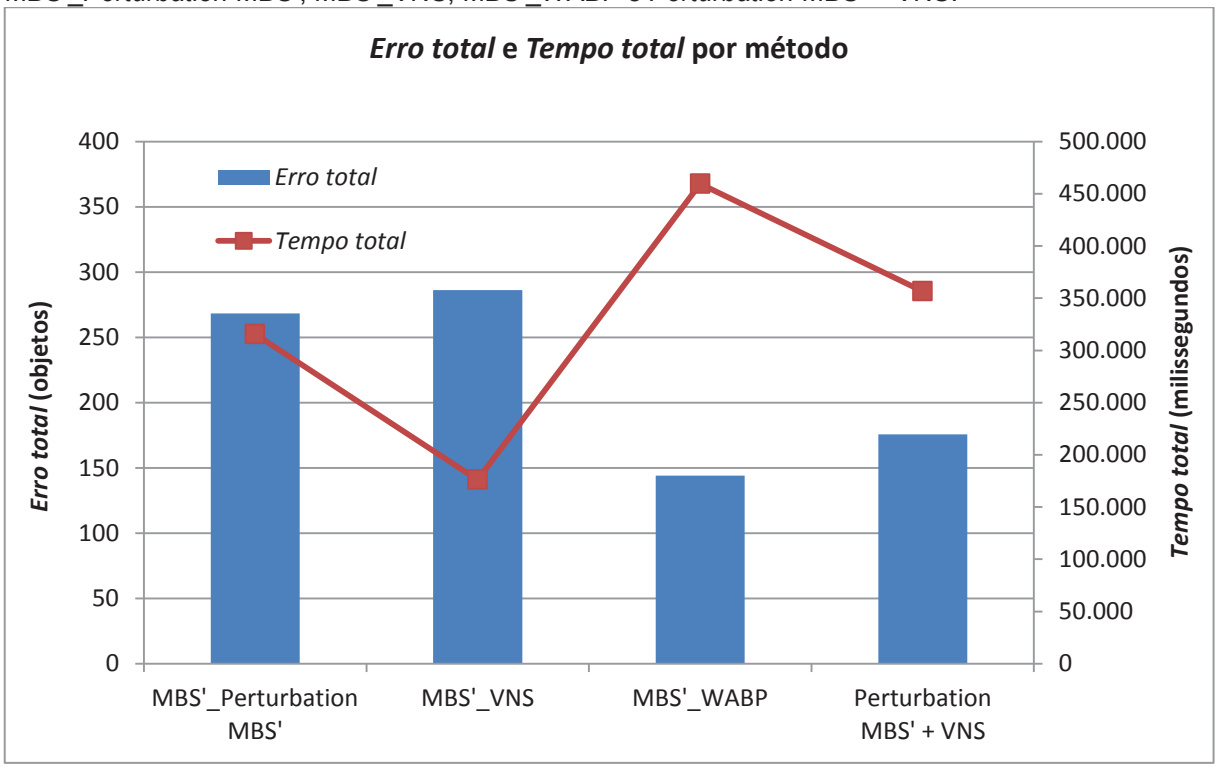
6.4.1 *Perturbation* MBS' + VNS

A combinação dos métodos VNS e *Perturbation* MBS' foi proposta por Fleszar e Hindi (2002) de modo a aliar a capacidade do método *Perturbation* MBS' em resolver os problemas do tipo *t* de forma eficiente ao melhoramento da solução nos conjuntos tipo *u*, DT1 e DT2, conseguido pelo método VNS. O método consiste em, inicialmente, aplicar o método *Perturbation* MBS' utilizando como solução inicial a heurística MBS'. A solução encontrada é então utilizada como solução inicial para o método VNS. Se em qualquer etapa o limitante inferior é atingido, tem-se a solução ótima e o processo é interrompido. Quando a quantidade de objetos na solução ótima é maior que o limitante inferior o método é forçado a executar todas as etapas de forma completa, prejudicando o tempo computacional para resolução. Os parâmetros dos métodos envolvidos neste processo são os mesmos testados nos métodos isolados, ou seja, 1000 iterações no método *Perturbation* MBS' e VNS explorando até 20 vizinhanças (*kmax* = 20). Os resultados apresentados na tabela 36 do anexo A se referem à média de 10 execuções do método para cada problema.

No gráfico 19 o método híbrido é comparado com os métodos MBS'_*Perturbation* MBS', MBS'_VNS e MBS'_WABP. Apesar da qualidade de solução superior aos métodos MBS'_*Perturbation* MBS' e MBS'_VNS, o método híbrido proposto por Fleszar e Hindi (2002) não foi capaz de superar os resultados obtidos pelo método MBS'_WABP. Apesar de obter a melhor qualidade de solução, o método MBS'_WABP teve pior desempenho em *Tempo total* entre os métodos

analisados no gráfico 19, registrando *Tempo total* de 459,6 segundos, ou seja, 28,9% superior ao método híbrido de Fleszar e Hindi.

Gráfico 19 – *Erro total* e *Tempo total* para resolução de todos os problemas – métodos MBS'_*Perturbation* MBS', MBS'_VNS, MBS'_WABP e *Perturbation* MBS' + VNS.



Fonte: Autor, 2014.

6.4.2 Nova proposta de hibridização

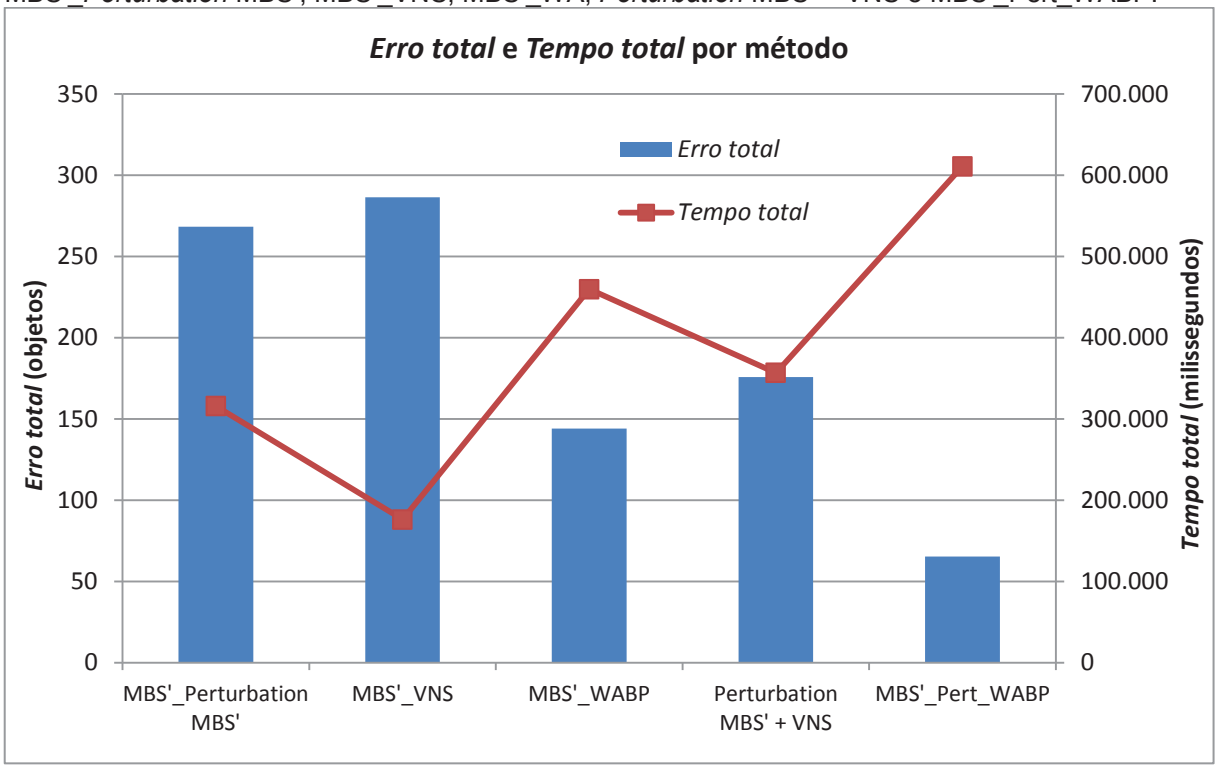
A partir dos resultados e conclusões obtidos anteriormente, é proposto neste trabalho um novo método híbrido, seguindo as ideias do método híbrido *Perturbation* MBS' + VNS proposto por Fleszar e Hindi (2002).

Ambos os métodos, WABP e VNS, procuram soluções ótimas locais com base em processos de troca. No método WABP, além das trocas utilizadas no método VNS, Loh *et al*. (2008) acrescentam ainda dois novos processos de troca (*Swap* (1,2) e *Swap* (2,2)) com objetivo de explorar de forma mais eficiente a vizinhança de soluções. A cada iteração são avaliadas todas as trocas possíveis para todos os pares de objetos, enquanto no método VNS a escolha das trocas acontece de forma aleatória. Conclui-se que o método WABP explora a vizinhança de soluções de forma mais eficaz em detrimento do tempo computacional. A hibridização proposta neste trabalho consiste na seguinte sequencia de

procedimentos: primeiramente, uma solução inicial é construída com o método MBS'. Caso o limitante inferior não seja atingido nesta etapa, a solução sofre uma perturbação pelo método *Perturbation* MBS'. Se até esta etapa o limitante inferior não foi atingido, um refinamento da solução é realizado pelo método de busca local WABP. O número de tentativas definido no método *Perturbation* MBS' é de 1.000 iterações enquanto a busca local WABP executa 50 iterações, com temperatura inicial T = 1, escala K = 0,05 e fator de redução de temperatura *Tred* = 0,95. Esta hibridização será denominada MBS'_Pert_WABP. Cada conjunto de problemas foi testado 10 vezes e os resultados, apresentados na tabela 37 do Anexo A, referem-se à média dos 10 testes.

O novo método híbrido foi bastante superior a todos os métodos analisados neste trabalho em termos de qualidade de solução, registrando *Erro total* médio de apenas 65,3 objetos, 54,6% melhor em relação ao segundo melhor método neste critério, o método MBS'_WABP, com *Erro total* de 144 objetos. Ainda, o método encontrou, em dois problemas do conjunto *u*120, soluções melhores que a melhor solução conhecida na literatura, além de atingir a melhor solução conhecida em 95,2% dos problemas, com quantidade média de *Sucessos* de 1295,5 problemas. No gráfico 20 são apresentados o *Erro total* e *Tempo total* dos 5 métodos que produziram melhor qualidade de solução.

Gráfico 20 – *Erro total* e *Tempo total* para resolução de todos os problemas – métodos MBS'_*Perturbation* MBS', MBS'_VNS, MBS'_WA, *Perturbation* MBS' + VNS e MBS'_Pert_WABP.

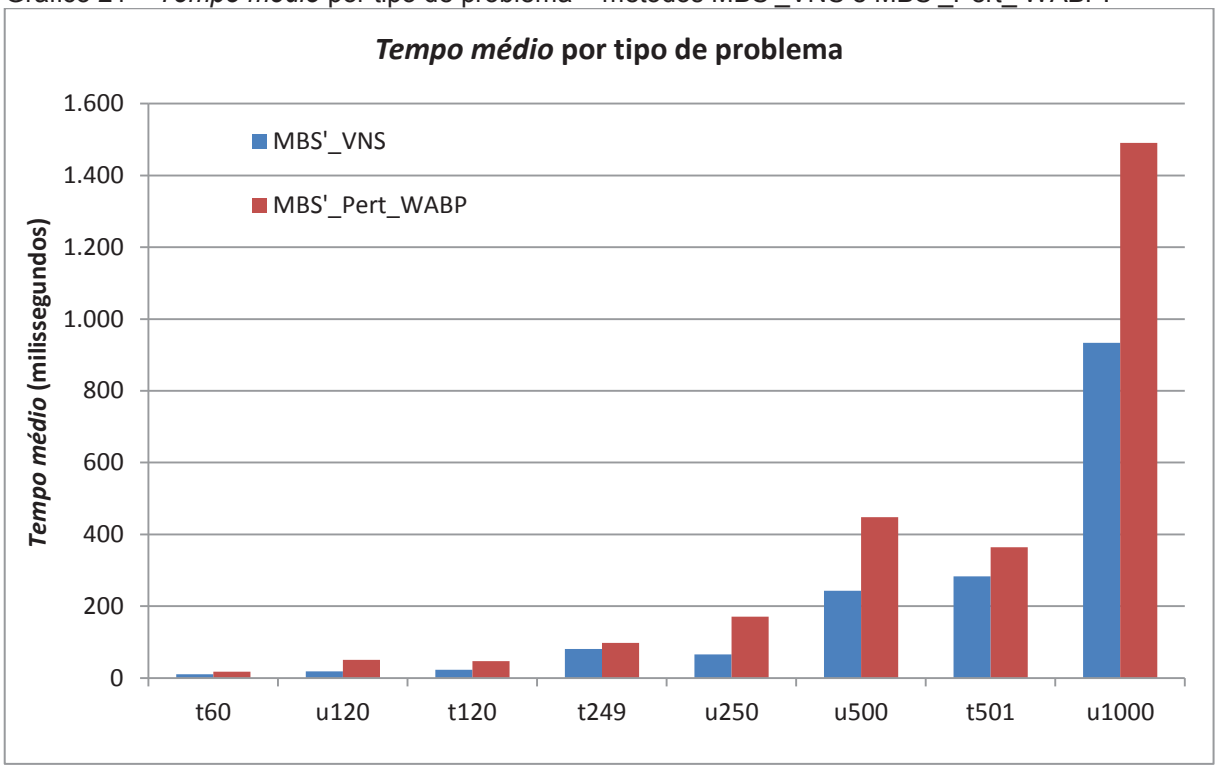


Fonte: Autor, 2014.

O método MBS'_Pert_WABP apresentou o pior desempenho computacional entre os métodos analisados no gráfico 20, sendo que o *Tempo médio* para resolução de um problema varia de aproximadamente 0,05 segundo para resolução de problemas de 60 itens até 1,49 segundo para problemas de 1.000 itens, como pode ser observado no gráfico 21, que apresenta uma comparação do *Tempo médio* por tipo de problema entre o método MBS'_Pert_WABP e o método VNS, mais eficiente dos métodos mostrados no gráfico 20 em relação ao tempo computacional. Apesar de uma diferença significativa de tempo computacional em favor do método MBS'_VNS quando comparado ao MBS'_Pert_WABP, este resolve os problemas de maior tamanho analisados neste trabalho (tipo *u*1000) em *Tempo médio* de 1,4 segundo, sendo que os demais problemas, com 501 itens ou menos, são resolvidos em *Tempo médio* inferior a 0,5 segundo.

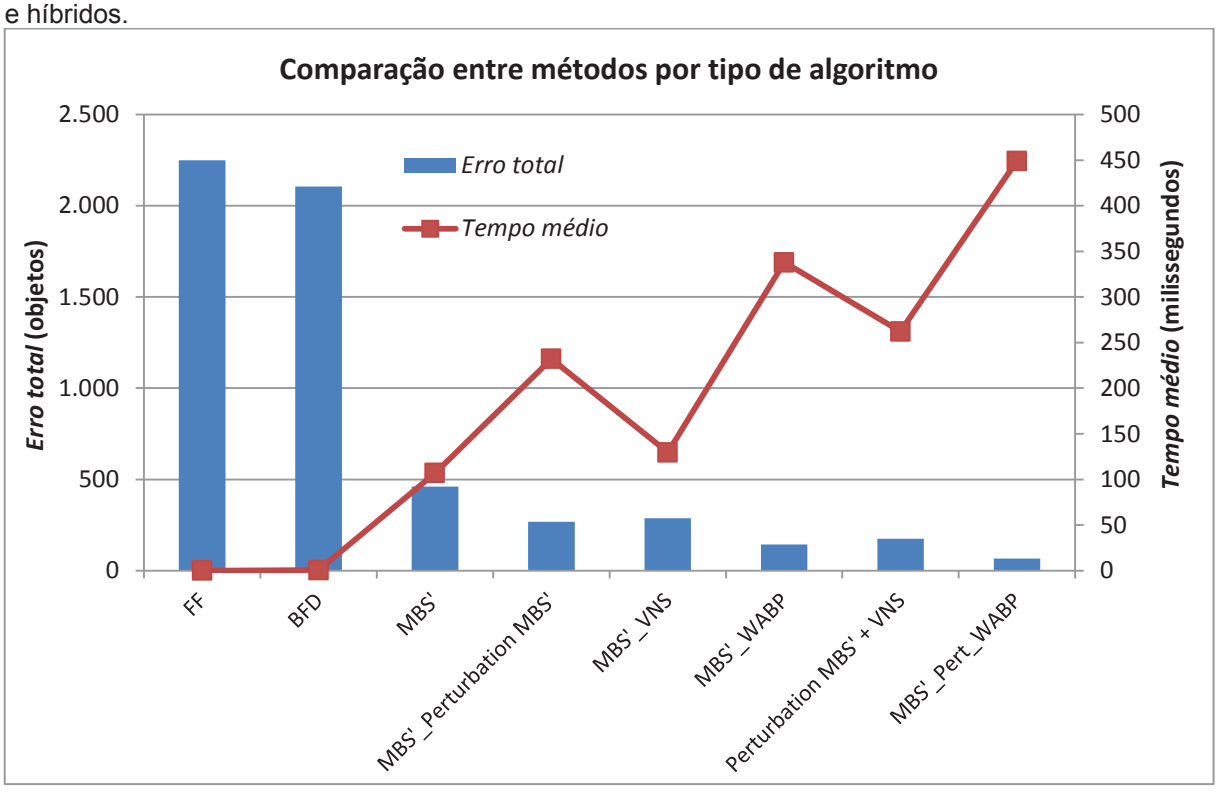
No gráfico 22 é apresentada uma comparação entre os métodos mais eficientes e mais eficazes, nesta ordem, para cada tipo de algoritmo: de aproximação; baseado em MBS; baseado em busca local; e híbrido.

Gráfico 21 – *Tempo médio* por tipo de problema – métodos MBS'_VNS e MBS'_Pert_ WABP.



Fonte: Autor, 2014.

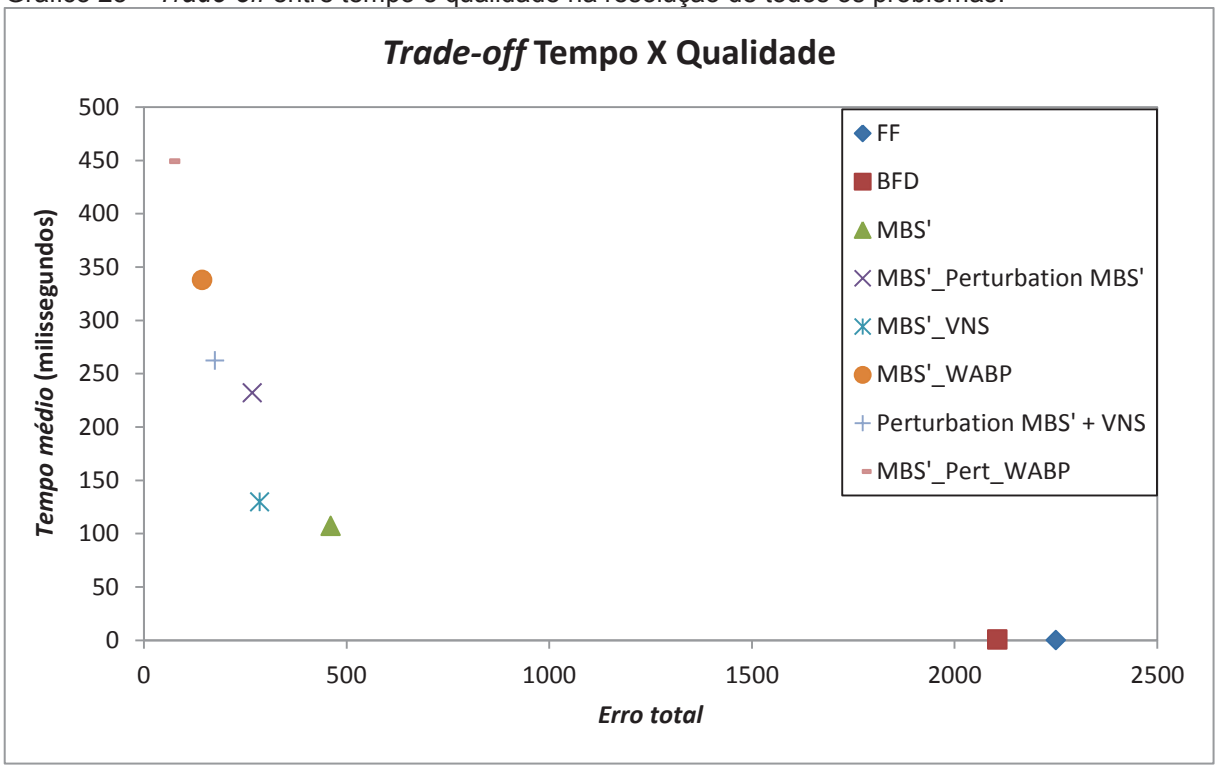
Gráfico 22 – Comparação entre métodos baseados em algoritmos de aproximação, MBS, busca local e híbridos.



Fonte: Autor, 2014.

Por fim, no gráfico 23 é mostrada a relação entre tempo computacional e qualidade de solução dos respectivos métodos. À medida que se caminha em direção a soluções melhores (*Erro total* = 0), há um incremento no tempo computacional.

Gráfico 23 – *Trade-off* entre tempo e qualidade na resolução de todos os problemas.



Fonte: Autor, 2014.

A tabela 5 mostra os cinco melhores métodos em cada conjunto de problemas em relação ao *Erro total* e ao *Tempo total*. Apesar de o método MBS'_Pert_WABP apresentar melhor resultado de *Erro total* em todos os conjuntos, este método apresenta melhores resultados apenas nos conjuntos *u*1000 e DT2. O método MBS'_Pert_WABP não foi classificado entre os cinco métodos com menor tempo computacional em nenhum conjunto de problemas. Os métodos de aproximação (FF, FFD, BF, BFD, WF e WFD) foram excluídos desta comparação por serem frequentemente utilizados como solução inicial para outros métodos, evitando desta forma distorção da análise.

Tabela 5 – Classificação dos métodos por conjunto de problemas.

Conjunto	Critério	1º (Melhor)	2º	3º	4º	5º
u120	***Erro total***	MBS'_WABP	**MBS'_Pert_WABP**	FF_WABP	BF_WABP	FFD_WABP
	Tempo total	BFD_VNS	FFD_VNS	MBS'	MBS	BFD_WABP
u250	***Erro total***	MBS'_WABP	**MBS'_Pert_WABP**	MBS_WABP	FF_WABP	FFD_WABP
	Tempo total	FFD_VNS	BFD_VNS	MBS'	MBS	MBS'_VNS
u500	***Erro total***	MBS'_WABP	MBS_WABP	**MBS'_Pert_WABP**	FFD_WABP	BFD_WABP
	Tempo total	FFD_VNS	BFD_VNS	FFD_Perturbation MBS'	BFD_Perturbation MBS'	MBS'
u1000	***Erro total***	**MBS'_Pert _WABP**	MBS'_WABP	FFD_WABP	BFD_WABP	MBS_WABP
	Tempo total	FFD_Perturbation MBS'	BFD_Perturbation MBS'	FFD_VNS	BFD_VNS	MBS'
t60	***Erro total***	WF_Perturbation MBS'	Perturbation MBS' + VNS	BFD_Perturbation MBS'	FF_Perturbation MBS'	FFD_Perturbation MBS'
	Tempo total	MBS'	MBS	MBS_VNS	MBS'_VNS	WF_Perturbation MBS'
t120	***Erro total***	BF_Perturbation MBS'	**MBS'_Pert_WABP**	Perturbation MBS' + VNS	FF_Perturbation MBS'	WF_Perturbation MBS'
	Tempo total	MBS'	MBS	MBS_VNS	MBS'_VNS	BF_Perturbation MBS'
t249	***Erro total***	FF_Perturbation MBS'	**MBS'_Pert_WABP**	Perturbation MBS' + VNS	WF_Perturbation MBS'	BF_Perturbation MBS'
	Tempo total	MBS'	MBS	FF_Perturbation MBS'	WF_Perturbation MBS'	BF_Perturbation MBS'
t501	***Erro total***	BF_Perturbation MBS'	**MBS'_Pert_WABP**	WF_Perturbation MBS'	MBS'_Perturbation MBS'	MBS_Perturbation MBS'
	Tempo total	BF_Perturbation MBS'	WF_Perturbation MBS'	FF_Perturbation MBS'	MBS'	MBS
DT1	***Erro total***	MBS'_WABP	**MBS'_Pert_WABP**	BF_WABP	BFD_WABP	FF_WABP
	Tempo total	BF_VNS	BFD_VNS	FFD_VNS	FF_VNS	MBS'
DT2	***Erro total***	**MBS'_Pert_WABP**	MBS'_WABP	MBS_WABP	Perturbation MBS' + VNS	MBS'_VNS
	Tempo total	MBS'	FF_Perturbation MBS'	BFD_Perturbation MBS'	FFD_Perturbation MBS'	BF_Perturbation MBS'
Total	***Erro total***	**MBS'_Pert_WABP**	MBS'_WABP	Perturbation MBS' + VNS	MBS'_Perturbation MBS'	MBS'_VNS
	Tempo total	MBS'	MBS'_VNS	MBS	MBS_VNS	FFD_Perturbation MBS'

Fonte: Autor, 2014.

Neste trabalho foram analisados um total 37 algoritmos, considerando cada método e suas diferentes configurações. Um resumo dos resultados obtidos por cada algoritmo na resolução dos 1.360 problemas é apresentado na tabela 6. São mostrados os resultados dos critérios: *Sucessos*; *Erro total*; *Erro total %; Erro médio*; *Erro máximo*; *Tempo total*; *Tempo médio* e *Tempo máximo*. Para cada critério são destacados os cinco melhores (negrito) e os cinco piores (vermelho) resultados.

Para os métodos que têm componentes aleatórios em seu procedimento, os resultados são apresentados como média de 10 resultados.

Tabela 6 – Classificação dos métodos por conjunto de problemas.

Método	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
FF	782	2.250	1,88%	1,7	24	73	0,1	2
FFD	788	2.107	1,76%	1,5	24	761	0,6	8
BF	783	2.195	1,83%	1,6	23	151	0,1	2
BFD	789	2.106	1,76%	1,5	24	781	0,6	9
WF	655	3.851	3,21%	2,8	66	131	0,1	2
WFD	656	2.361	1,97%	1,7	24	780	0,6	15
MBS	679	1.341	1,12%	1,0	9	192.665	141,7	15.357
MBS'	1.057	461	0,38%	0,3	9	145.510	107,0	7.294
FF_Perturbation MBS'	998,1	795,9	0,66%	0,6	14,3	264.960,5	194,8	3.786,9
FFD_Perturbation MBS'	988,1	678,6	0,57%	0,5	15,7	220.910,5	162,4	3.930,0
BF_Perturbation MBS'	1.001,0	783,3	0,65%	0,6	14,5	266.040,3	195,6	3.963,1
BFD_Perturbation MBS'	992,8	658,2	0,55%	0,5	14,7	224.651,1	165,2	3.784,4
WF_Perturbation MBS'	894,5	980,7	0,82%	0,7	14,9	439.675,6	323,3	3.966,8
WFD_Perturbation MBS'	869,6	950,7	0,79%	0,7	15,4	426.519,7	313,6	5.505,9
MBS_Perturbation MBS'	791,2	1.175,0	0,98%	0,9	9,0	358.821,7	263,8	14.308,6
MBS'_Perturbation MBS'	1.159,2	268,3	0,22%	0,2	7,5	315.622,8	232,1	7.178,7
FF_VNS	979,9	523,0	0,44%	0,4	7,3	392.177,6	288,4	12.455,7
FFD_VNS	980,8	515,5	0,43%	0,4	6,7	267.123,7	196,4	7.858,9
BF_VNS	979,4	529,1	0,44%	0,4	6,9	372.504,5	273,9	12.069,3
BFD_VNS	975,0	518,9	0,43%	0,4	6,7	269.624,5	198,3	7.461,7
WF_VNS	885,3	641,1	0,53%	0,5	7,5	709.656,6	521,8	28.728,7
WFD_VNS	878,7	641,1	0,53%	0,5	6,7	362.979,2	266,9	7.668,9
MBS_VNS	867,6	780,0	0,65%	0,6	8,5	220.674,8	162,3	14.412,4
MBS'_VNS	1.103,4	286,3	0,24%	0,2	3,8	176.131,0	129,5	7.185,5
FF_WABP	1.125	322	0,27%	0,2	9	500.737	368,2	6.095
FFD_WABP	1.140	418	0,35%	0,3	9	429.051	315,5	5.836
BF_WABP	1.122	317	0,26%	0,2	9	504.414	370,9	6.639
BFD_WABP	1.142	416	0,35%	0,3	9	423.981	311,8	5.730
WF_WABP	1.097	470	0,39%	0,3	16	541.197	397,9	6.417
WFD_WABP	1.105	472	0,39%	0,3	9	499.987	367,6	6.148
MBS_WABP	938	606	0,51%	0,4	6	515.982	379,4	14.170
MBS'_WABP	1.219	144	0,12%	0,1	2	459.600	337,9	7.212
FFD_AugNN	972,7	1.181,7	0,99%	0,9	17,0	1.673.666,1	1.230,6	16.791,4
BFD_AugNN	959,1	1.062,0	0,89%	0,8	15,0	3.295.920,8	2.423,5	32.918,9
WFD_AugNN	766,0	1.453,0	1,21%	1,1	13,8	5.347.162,3	3.931,7	51.328,5
Perturbation MBS' + VNS	1.200,1	175,7	0,15%	0,1	3,0	356.627,8	262,2	7.138,1
MBS_Pert_WABP	1.295,5	65,3	0,05%	0,0	1,1	610.787,2	449,1	7.172,6

7 CONCLUSÕES E DIRECIONAMENTOS

Neste trabalho foram avaliados alguns métodos existentes na literatura para resolução de problemas do tipo BP, uma classe particular dos PCEs. Também foram testadas algumas variações dos métodos, quando aplicável, de forma a avaliar como cada um se comporta no tratamento dos problemas nos quais foram testados. Os problemas testados são encontrados na literatura e possuem diferentes características, que conferem diferentes graus de dificuldade na obtenção de soluções. Foi possível observar que alguns métodos podem ser bastante eficazes na resolução de um tipo de problema e ser pouco eficazes na resolução de outro, como no caso do método *Perturbation* MBS', que foi o único capaz de encontrar respostas ótimas nos conjuntos do tipo *t*, ou do método WABP que foi bastante eficaz na resolução dos problemas do tipo *u*, apesar de não encontrar nenhuma resposta ótima nos conjuntos do tipo *t*. Nos métodos de aproximação, as variações onde os itens são previamente classificados em ordem decrescente produzem resultados superiores, entretanto, nos problemas do tipo *t*, os métodos sem classificação apresentam melhores resultados.

A análise dos métodos existentes e suas variações forneceu conclusões importantes para a determinação dos procedimentos que foram combinados para construção do método híbrido, MBS'_Pert_WABP. Este método se aproveita da efetividade da solução inicial obtida pelo método MBS', do melhoramento da solução conseguido pelo procedimento de perturbação da solução *Perturbation* MBS' e da capacidade de refinamento da solução pela busca local realizada pelo método WABP.

Também foi determinante a utilização de diversos problemas com características diferentes para testes dos métodos avaliados. Utilização limitada de tipos de problemas poderia alterar o resultado da composição do método híbrido, ou até mesmo não justificar sua utilização. O método híbrido proposto é capaz de resolver todos os tipos de problemas avaliados neste trabalho de forma eficaz, mas não é possível garantir a eficácia em outros tipos de problema. Testes em problemas reais e também em outros tipos de problemas poderiam confirmar a robustez do método.

Os problemas de BP analisados são caracterizados por grande heterogeneidade de itens, entretanto, o método MBS'_Pert_WABP pode ser utilizado

para resolução de problemas de Corte de Estoque, caracterizados por pouca heterogeneidade de itens. Uma desvantagem na utilização deste método em problemas de Corte de Estoque, e a dificuldade para agrupamento de padrões de corte, visto que, dependendo do processo de corte utilizado, a padronização e agrupamento de padrões de corte pode trazer ganhos significativos de produtividade, ganhos que podem inclusive se sobrepor ao ganho em economia de matéria-prima.

Os parâmetros utilizados no método MBS'_Pert_WABP apresentado neste trabalho foram os mesmos utilizados nos métodos originais que compõem o híbrido, deixando uma oportunidade para otimização de parâmetros. É possível que o conjunto de parâmetros utilizado não seja o mais eficiente devido à combinação dos algoritmos. Um trabalho de refinamento do método pode revelar possíveis ganhos de sinergia, decorrentes desta combinação, que resultem em maior eficiência e eficácia do método MBS'_Pert_WABP.

Como o sistema de trocas utilizado no método WABP avalia uma função objetivo, a resolução de um problema multicritério poderia ser testada. As duas primeiras etapas do método híbrido tratariam a minimização de objetos e na etapa de refinamento da solução com WABP, outro critério poderia ser avaliado para otimização.

O método MBS'_Pert_WABP apresentado neste trabalho superou todos demais métodos avaliados neste trabalho em qualidade de solução. O erro médio para cada problema é de 0,05 objetos. No conjunto dos 1.360 problemas analisados, a melhor resposta conhecida foi encontrada em aproximadamente 95% dos casos, sendo que, em dois problemas, o método foi capaz de encontrar soluções melhores que a melhor solução conhecida na literatura. Os resultados obtidos por este método representariam uma perda de matéria-prima de 0,05%, desconsiderando perdas de corte relativas a restrições de máquina, contra uma perda de 0,15% do método híbrido encontrado na literatura (*Perturbation* MBS' + VNS).

A aplicação método MBS'_Pert_WABP reduziu o *erro total* na resolução de todos os problemas em 54,6% em relação ao segundo melhor método avaliado neste trabalho. O segundo melhor método em qualidade de solução, MBS'_WABP, também é produto deste trabalho, resultando do teste de diferentes configurações do método WABP, sendo que Loh *et al*. (2008), que apresentaram esta meta-heurística para resolução de problemas de BP, propõem como solução inicial para o WABP a

heurística FFD.

Comparando o método híbrido proposto neste trabalho ao método mais eficaz encontrado na literatura avaliado neste trabalho (*Perturbation* MBS' + VNS), a redução no erro total é da ordem de 62,8%.

Dadas as conclusões, propõe-se alguns direcionamentos futuros:

- Refinamento do método híbrido MBS'_Pert_WABP na busca de parâmetros que otimizem as respostas do algoritmo tanto em tempo computacional como em qualidade de solução;

- Aplicação do método MBS'_Pert_WABP em outros conjuntos de problemas do tipo BP, além da análise dos resultados em problemas reais;

- Adaptação e aplicação do método em problemas do tipo Corte de Estoque. Comparação com métodos tradicionais para resolução de problemas desta categoria.

REFERÊNCIAS

AGARWAL, A. Theoretical insights into the augmented-neural-network approach for combinatorial optimization, **Annals of Operations Research**, v. 168, p. 101-117, 2009.

AGARWAL, A.; PIRKULI, H.; JACOB, V. S. Augmented neural networks for task scheduling, **European Journal of Operational Research**, v. 151, p. 481-502, 2003.

ALMEIDA, R.; STEINER, M. T. A. Aplicação de uma rede neural aumentada ajustada por delineamento de experimentos para resolução de problemas de corte e empacotamento, **in: XVI Simpósio de Pesquisa Operacional e Logística da Marinha**, Rio de Janeiro, v. 16, 2013a.

ALMEIDA, R.; STEINER, M. T. A. Aplicação de uma rede neural aumentada para resolução de problemas de corte e empacotamento utilizando novas estratégias de aprendizagem, **in: XLV Simpósio Brasileiro de Pesquisa Operacional**, Natal, v. 45, 2013b.

ARAUJO, S. A.; CONSTANTINO, A. A.; POLDI, K. C. An evolutionary algorithm for the one-dimensional cutting stock problem, **International Transactions in Operational Research**, v. 18, p. 115-127, 2010.

AYOB, M.; NAZRI, M. Z. A.; FEI, Y. X. Local search heuristics for the one dimensional bin packing problems, **Journal of Applied Sciences**, v. 13 (6), p. 919-923, 2013.

CARVALHO, J. M. V. Exact solution of bin-packing problems using column generation and branch-and-bound, **Annals of Operation Research**, v. 86, p. 629-659, 1999.

CARVALHO, J. M. V. LP models for bin packing and cutting stock problems, **European Journal of Operational Research**, v. 141, p. 253-273, 2002.

CHEN, C. S.; HART, S. M.; THAM, W. M. A simulated annealing heuristic for the one-dimensional cutting stock problem, **European Journal of Operations Research**, v. 93, p. 522-535, 1996.

CUI, Y.; YANG, Y. A heuristic for the one-dimensional cutting stock problem with usable leftover, **European Journal of Operational Research**, v. 204, p. 245-250.

DYCKHOFF, H. A new linear programming approach to the cutting stock problem, **Operations Research**, v. 29, p. 1092-1104, 1981.

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

EISEMANN, K. The trim problem, **Management Science**, v. 3, p. 279-284, 1957.

EISENBRAND, F.; SHMONIN, G. Carathéodory bounds for integer cones, **Operations Research Letters**, v. 34, p. 564-568, 2006.

FALKENAUER, E. A hybrid grouping genetic algorithm for bin packing, **Journal of Heuristics**, v. 2, p. 5-30, 1996.

FLESZAR, K.; HINDI, K. S. New heuristics for one-dimensional bin-packing, **Computers & Operations Research**, v. 29, p. 821-839, 2002.

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

GILMORE, P. C.; GOMORY, R. E. A linear programming approach to the cutting-stock problem – Part II, **Operations Research**, v. 11, p. 864-888, 1963.

GUPTA J. N. D.; HO, J. C. A new heuristic algorithm for the one-dimensional bin-packing problem, **Production Planning & Control**, v. 10, p. 598-603, 1999.

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

JANSEN, K.; SOLIS-OBA, R. An OPT+1 algorithm for cutting stock problem with a constant number of object lengths, **in: Proceedings of the 14th International Conference on Integer Programming and Combinatorial Optimization,** v. 6080, p. 438-449, 2010.

JANSEN, K.; SOLIS-OBA, R. A simple OPT+1 algorithm for cutting stock under the modified integer round-up property assumption, **Information Processing Letters**, v. 111, p. 479-482, 2011.

JOHNSON, D. S. Approximation algorithms for combinatorial problems, **Journal of Computer and System Sciences**, v. 9, p. 256-278, 1974.

KANTOROVICH, L. V. Mathematical Methods of Organizing and Planning Production, 1939 (Versão em lingua inglesa publicada em **Management Science**, v. 6, p. 366-422, 1960).

KASAP, N.; AGARWAL, A. Augmented neural networks and problem structure-based heuristics for the bin-packing problem, **International Journal of Systems Science**, v. 43, p. 1412-1430, 2012.

LOH, K.; GOLDEN, B.; WASIL, E. Solving the one-dimensional bin packing problem with a weight annealing heuristic, **Computers & Operations Research**, v. 35, p. 2283-2291, 2008.

MARTELLO, S.; TOTH, P. Lower bounds and reduction procedures for the bin packing problem, **Discrete Applied Mathematics**, v. 11, p. 59-70, 1990a.

MARTELLO, S.; TOTH, P. **Knapsack Problems: Algorithms and Computer Implementations**, John Wiley & Sons, Cichester, 1990b.

MLADENOVIC, N.; HANSEN P. Variable neighborhood search, **Computers & Operations Research**, v. 24, n. 11, p. 1097-1100, 1997.

MOSQUERA, G. P. **Contribuições para o Problema de Corte de Estoque Bidimensional na Indústria Moveleira**, 146 p., Dissertação (Mestrado em Matemática Aplicada) – Universidade Estadual Paulista, SP, 2007.

NINIO, M.; SCHNEIDER, J. J. Weight annealing, **Physica A**, v. 349, p. 649-666, 2005.

PILEGGI, G. C. F.; MORABITO, R.; ARENALES, M. N. Abordagens para otimização integrada dos problemas de geração e sequenciamento de padrões de corte: caso unidimensional, **Pesquisa Operacional**, v. 25, n. 3, p. 417-447, 2005.

POLDI, K. C.; ARENALES, M. N. O problema de corte de estoque unidimensional multiperíodo, **Pesquisa Operacional**, v. 30, n. 1, p. 153-174, 2010.

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, 627-645, 1997.

SIMCHI-LEVI, D. New worst-case results for the bin-packing problem, **Naval Research Logistics**, v. 41, p. 579-585, 1994.

STADLER, H. A comparison of two optimization procedures for 1- and 1.1/2-dimensional cutting stock problems, **OR Spectrum**, v. 10, p. 97-111, 1988.

VASKO, F. J. *et al*. A real-time one-dimensional cutting stock algorithm for balanced cutting patterns, **Operations Research Letters**, v. 14, p. 275-282, 1993.

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

WOODCOCK, A. J.; WILSON, J. M. A hybrid tabu search/branch & bound approach to solving the generalized assignment problem, **European Journal of Operations Research**, v. 207, p. 566-578, 2010.

ANEXO A – TABELAS DE RESULTADO POR MÉTODO

Tabela 7 – Resultados FF.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	0	63	6,41%	3,2	5	< 0,49	< 0,024	< 0,49
u250	20	0	128	6,29%	6,4	9	< 0,49	< 0,024	< 0,49
u500	20	0	231	5,74%	11,6	15	< 0,49	< 0,024	< 0,49
u1000	20	0	419	5,23%	21,0	24	28	1,4	2
t60	20	0	31	7,75%	1,6	2	< 0,49	< 0,024	< 0,49
t120	20	0	57	7,13%	2,9	4	< 0,49	< 0,024	< 0,49
t249	20	0	100	6,02%	5,0	6	< 0,49	< 0,024	< 0,49
t501	20	0	190	5,69%	9,5	11	< 0,49	< 0,024	< 0,49
DT1	720	546	283	0,36%	0,4	5	45	0,1	1
DT2	480	236	748	3,69%	1,6	21	< 0,49	< 0,024	< 0,49
Totais	1.360	782	2.250	1,88%	1,7	24	73	0,1	2

Fonte: Autor, 2014.

Tabela 8 – Resultados FFD.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	6	14	1,42%	0,7	1	< 0,49	< 0,024	< 0,49
u250	20	0	28	1,38%	1,4	3	< 0,49	< 0,024	< 0,49
u500	20	0	54	1,34%	2,7	3	40	2,0	2
u1000	20	0	97	1,21%	4,9	7	150	7,5	8
t60	20	0	64	16,00%	3,2	4	< 0,49	< 0,024	< 0,49
t120	20	0	116	14,50%	5,8	7	< 0,49	< 0,024	< 0,49
t249	20	0	240	14,46%	12,0	13	< 0,49	< 0,024	< 0,49
t501	20	0	463	13,86%	23,2	24	40	2,0	2
DT1	720	546	283	0,36%	0,4	5	380	0,5	4
DT2	480	236	748	3,69%	1,6	21	151	0,3	3
Totais	1.360	788	2.107	1,76%	1,5	24	761	0,6	8

Fonte: Autor, 2014.

Tabela 9 – Resultados BF.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	0	57	5,80%	2,9	4	< 0,49	< 0,024	< 0,49
u250	20	0	120	5,90%	6,0	8	1	0,1	1
u500	20	0	216	5,37%	10,8	13	2	0,1	1
u1000	20	0	396	4,94%	19,8	23	40	2,0	2
t60	20	0	31	7,75%	1,6	2	< 0,49	< 0,024	< 0,49
t120	20	0	56	7,00%	2,8	4	< 0,49	< 0,024	< 0,49
t249	20	0	100	6,02%	5,0	6	< 0,49	< 0,024	< 0,49
t501	20	0	189	5,66%	9,5	12	< 0,49	< 0,024	< 0,49
DT1	720	547	282	0,36%	0,4	5	108	0,2	2
DT2	480	236	748	3,69%	1,6	21	< 0,49	< 0,024	< 0,49
Totais	1.360	783	2.195	1,83%	1,6	23	151	0,1	2

Fonte: Autor, 2014.

Tabela 10 – Resultados BFD.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	6	14	1,42%	0,7	1	< 0,49	< 0,024	< 0,49
u250	20	0	28	1,38%	1,4	3	< 0,49	< 0,024	< 0,49
u500	20	0	54	1,34%	2,7	3	40	2,0	2
u1000	20	0	97	1,21%	4,9	7	164	8,2	9
t60	20	0	64	16,00%	3,2	4	< 0,49	< 0,024	< 0,49
t120	20	0	116	14,50%	5,8	7	< 0,49	< 0,024	< 0,49
t249	20	0	240	14,46%	12,0	13	< 0,49	< 0,024	< 0,49
t501	20	0	463	13,86%	23,2	24	40	2,0	2
DT1	720	547	282	0,36%	0,4	5	386	0,5	2
DT2	480	236	748	3,69%	1,6	21	151	0,3	2
Totais	1.360	789	2.106	1,76%	1,5	24	781	0,6	9

Fonte: Autor, 2014.

Tabela 11 – Resultados WF.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	0	150	15,26%	7,5	12	< 0,49	< 0,024	< 0,49
u250	20	0	308	15,14%	15,4	21	< 0,49	< 0,024	< 0,49
u500	20	0	590	14,66%	29,5	36	2	0,1	1
u1000	20	0	1.143	14,27%	57,2	66	40	2,0	2
t60	20	0	38	9,50%	1,9	3	< 0,49	< 0,024	< 0,49
t120	20	0	57	7,13%	2,9	4	< 0,49	< 0,024	< 0,49
t249	20	0	111	6,69%	5,6	8	< 0,490	< 0,024	< 0,49
t501	20	0	205	6,14%	10,3	13	< 0,49	< 0,024	< 0,49
DT1	720	442	465	0,59%	0,6	5	89	0,1	2
DT2	480	213	784	3,87%	1,6	21	< 0,49	< 0,024	< 0,49
Totais	1.360	655	3.851	3,21%	2,8	66	131	0,1	2

Fonte: Autor, 2014.

Tabela 12 – Resultados WFD.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	1	22	2,24%	1,1	2	1	0,1	1
u250	20	0	34	1,67%	1,7	3	< 0,49	< 0,024	< 0,49
u500	20	0	61	1,52%	3,1	5	40	2,0	2
u1000	20	0	112	1,40%	5,6	9	160	8,0	8
t60	20	0	64	16,00%	3,2	4	< 0,49	< 0,024	< 0,49
t120	20	0	116	14,50%	5,8	7	< 0,49	< 0,024	< 0,49
t249	20	0	240	14,46%	12,0	13	< 0,49	< 0,024	< 0,49
t501	20	0	463	13,86%	23,2	24	40	2,0	2
DT1	720	442	465	0,59%	0,6	5	389	0,5	15
DT2	480	213	784	3,87%	1,6	21	150	0,3	2
Totais	1.360	656	2.361	1,97%	1,7	24	780	0,6	15

Fonte: Autor, 2014.

Tabela 13 – Resultados MBS.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	12	10	1,02%	0,5	2	287	14,4	26
u250	20	10	12	0,59%	0,6	2	1.163	58,2	63
u500	20	11	12	0,30%	0,6	2	4.861	243,1	304
u1000	20	7	16	0,20%	0,8	3	18.334	916,7	1.035
t60	20	0	20	5,00%	1,0	1	62	3,1	4
t120	20	0	20	2,50%	1,0	1	242	12,1	13
t249	20	0	23	1,39%	1,2	2	1.020	51,0	53
t501	20	0	41	1,23%	2,1	3	4.136	206,8	224
DT1	720	252	1.057	1,35%	1,5	9	68.651	95,3	468
DT2	480	387	130	0,64%	0,3	5	93.909	195,6	15.357
Totais	1.360	679	1.341	1,12%	1,0	9	192.665	141,7	15.357

Fonte: Autor, 2014.

Tabela 14 – Resultados MBS'.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	11	9	0,92%	0,5	1	273	13,7	24
u250	20	14	7	0,34%	0,4	2	1.142	57,1	60
u500	20	11	12	0,30%	0,6	2	4.793	239,7	313
u1000	20	7	16	0,20%	0,8	3	18.154	907,7	1.042
t60	20	0	20	5,00%	1,0	1	60	3,0	3
t120	20	0	20	2,50%	1,0	1	224	11,2	12
t249	20	0	36	2,17%	1,8	3	991	49,6	53
t501	20	0	76	2,28%	3,8	7	3.969	198,5	203
DT1	720	633	101	0,13%	0,1	3	63.083	87,6	458
DT2	480	381	164	0,81%	0,3	9	52.821	110,0	7.294
Totais	1.360	1.057	461	0,38%	0,3	9	145.510	107,0	7.294

Fonte: Autor, 2014.

Tabela 15 – Resultados FF_*Perturbation* MBS' - Média de 10 testes.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	2,1	37,9	3,86%	1,9	3,9	1.758,0	87,9	164,8
u250	20	2,5	50,2	2,47%	2,5	5,9	3.912,0	195,6	295,6
u500	20	0,5	70,7	1,76%	3,5	7,8	10.943,0	547,2	805,8
u1000	20	0,7	109,1	1,36%	5,5	12,7	43.272,0	2.163,6	2.731,8
t60	20	19,5	0,5	0,13%	0,0	0,4	255,6	12,8	62,8
t120	20	19,0	1,1	0,14%	0,1	0,9	713,4	35,7	106,5
t249	20	19,4	0,9	0,05%	0,0	0,9	1.127,2	56,4	112,4
t501	20	18,0	3,4	0,10%	0,2	2,0	3.564,6	178,2	271,0
DT1	720	572,4	224,8	0,29%	0,3	5,0	146.431,0	203,4	2.455,6
DT2	480	344,0	297,3	1,47%	0,6	14,3	52.983,7	110,4	3.786,9
Totais	1.360	998,1	795,9	0,66%	0,6	14,3	264.960,5	194,8	3.786,9

Fonte: Autor, 2014.

Tabela 16 – Resultados FFD_*Perturbation* MBS' - Média de 10 testes.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	9,2	10,8	1,10%	0,5	1,0	677,6	33,9	85,1
u250	20	5,6	16,5	0,81%	0,8	2,0	1.393,4	69,7	128,9
u500	20	2,5	27,5	0,68%	1,4	3,0	2.712,4	135,6	203,4
u1000	20	0,1	55,9	0,70%	2,8	5,5	5.642,2	282,1	388,6
t60	20	19,5	0,5	0,13%	0,0	0,5	360,8	18,0	79,6
t120	20	18,8	1,7	0,21%	0,1	1,3	994,0	49,7	122,4
t249	20	11,5	13,1	0,79%	0,7	2,0	2.227,4	111,4	170,5
t501	20	4,6	24,4	0,73%	1,2	2,7	6.167,6	308,4	405,8
DT1	720	572,3	226,7	0,29%	0,3	5,0	147.033,1	204,2	2.441,8
DT2	480	344,0	301,5	1,49%	0,6	15,7	53.702,0	111,9	3.930,0
Totais	1.360	988,1	678,6	0,57%	0,5	15,7	220.910,5	162,4	3.930,0

Fonte: Autor, 2014.

Tabela 17 – Resultados BF_*Perturbation* MBS' - Média de 10 testes.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	1,7	36,2	3,68%	1,8	3,9	1.700,2	85,0	136,4
u250	20	1,3	50,2	2,47%	2,5	5,1	3.937,3	196,9	317,1
u500	20	1,3	69,6	1,73%	3,5	8,1	11.009,1	550,5	792,0
u1000	20	0,7	108,5	1,35%	5,4	12,0	42.480,1	2.124,0	2.823,7
t60	20	19,5	0,6	0,15%	0,0	0,6	235,6	11,8	56,5
t120	20	19,4	0,8	0,10%	0,0	0,6	698,1	34,9	95,2
t249	20	18,6	2,2	0,13%	0,1	1,2	1.216,1	60,8	119,2
t501	20	19,1	1,6	0,05%	0,1	1,3	3.487,2	174,4	249,2
DT1	720	575,5	222,0	0,28%	0,3	5,0	147.018,3	204,2	2.457,6
DT2	480	343,9	291,6	1,44%	0,6	14,5	54.258,3	113,0	3.963,1
Totais	1.360	1.001,0	783,3	0,65%	0,6	14,5	266.040,3	195,6	3.963,1

Fonte: Autor, 2014.

Tabela 18 – Resultados BFD_*Perturbation* MBS' - Média de 10 testes.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	9,0	11,0	1,12%	0,6	1,0	674,1	33,7	82,2
u250	20	6,0	15,9	0,78%	0,8	2,0	1.449,8	72,5	137,1
u500	20	2,6	26,8	0,67%	1,3	3,0	2.950,9	147,5	235,7
u1000	20	0,6	50,8	0,63%	2,5	5,4	6.027,7	301,4	450,5
t60	20	19,7	0,3	0,08%	0,0	0,3	296,4	14,8	55,2
t120	20	18,5	2,0	0,25%	0,1	1,2	979,4	49,0	112,0
t249	20	11,0	12,9	0,78%	0,6	2,0	2.304,6	115,2	187,3
t501	20	4,0	25,5	0,76%	1,3	2,6	6.366,0	318,3	420,2
DT1	720	576,9	222,3	0,28%	0,3	5,0	150.432,0	208,9	2.501,4
DT2	480	344,5	290,7	1,44%	0,6	14,7	53.170,2	110,8	3.784,4
Totais	1.360	992,8	658,2	0,55%	0,5	14,7	224.651,1	165,2	3.784,4

Fonte: Autor, 2014.

Tabela 19 – Resultados WF_*Perturbation* MBS' - Média de 10 testes.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	2,0	38,4	3,91%	1,9	3,7	2.016,6	100,8	174,2
u250	20	1,6	50,8	2,50%	2,5	5,7	4.538,9	226,9	357,3
u500	20	0,6	69,8	1,73%	3,5	8,0	12.205,9	610,3	886,7
u1000	20	0,1	118,6	1,48%	5,9	12,5	56.387,7	2.819,4	3.723,6
t60	20	19,8	0,2	0,05%	0,0	0,2	227,9	11,4	55,0
t120	20	19,2	1,3	0,16%	0,1	1,3	730,7	36,5	97,7
t249	20	19,0	1,5	0,09%	0,1	1,0	1.209,0	60,5	115,6
t501	20	18,8	2,1	0,06%	0,1	1,5	3.502,9	175,1	279,1
DT1	720	481,2	396,8	0,51%	0,6	5,0	294.186,4	408,6	3.966,8
DT2	480	332,2	301,2	1,49%	0,6	14,9	64.669,6	134,7	3.927,1
Totais	1.360	894,5	980,7	0,82%	0,7	14,9	439.675,6	323,3	3.966,8

Fonte: Autor, 2014.

Tabela 20 – Resultados WFD_*Perturbation* MBS' - Média de 10 testes.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	3,6	19,8	2,01%	1,0	2,0	1.412,4	70,6	124,1
u250	20	2,8	28,9	1,42%	1,4	3,0	3.089,7	154,5	258,0
u500	20	0,2	49,9	1,24%	2,5	4,9	9.070,5	453,5	795,7
u1000	20	0,0	98,6	1,23%	4,9	9,0	43.937,4	2.196,9	5.505,9
t60	20	19,2	0,9	0,23%	0,0	0,8	366,8	18,3	72,7
t120	20	18,1	3,1	0,39%	0,2	1,7	1.068,8	53,4	128,3
t249	20	11,1	13,5	0,81%	0,7	2,1	2.309,9	115,5	183,0
t501	20	3,7	26,5	0,79%	1,3	2,7	6.510,0	325,5	446,1
DT1	720	479,6	399,1	0,51%	0,6	5,0	294.030,4	408,4	4.057,1
DT2	480	331,3	310,4	1,53%	0,6	15,4	64.723,8	134,8	3.868,9
Totais	1.360	869,6	950,7	0,79%	0,7	15,4	426.519,7	313,6	5.505,9

Fonte: Autor, 2014.

Tabela 21 – Resultados MBS_*Perturbation* MBS' - Média de 10 testes.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	12,0	9,9	1,01%	0,5	2,0	753,3	37,7	89,9
u250	20	11,0	10,7	0,53%	0,5	2,0	1.790,1	89,5	155,7
u500	20	11,1	11,9	0,30%	0,6	2,0	5.378,8	268,9	362,6
u1000	20	8,1	14,8	0,18%	0,7	3,0	19.311,8	965,6	1.089,9
t60	20	19,5	1,3	0,31%	0,1	1,0	399,5	20,0	66,4
t120	20	18,8	1,7	0,21%	0,1	1,0	868,7	43,4	104,5
t249	20	18,2	2,3	0,14%	0,1	1,0	1.732,9	86,6	149,2
t501	20	18,6	2,1	0,06%	0,1	1,4	5.257,4	262,9	341,2
DT1	720	266,2	1.018,0	1,30%	1,4	9,0	204.499,1	284,0	2.945,1
DT2	480	407,7	102,4	0,51%	0,2	5,0	118.830,1	247,6	14.308,6
Totais	1.360	791,2	1.175,0	0,98%	0,9	9,0	358.821,7	263,8	14.308,6

Fonte: Autor, 2014.

Tabela 22 – Resultados MBS'_*Perturbation* MBS' - Média de 10 testes.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	11,2	8,8	0,90%	0,4	1,0	809,3	40,5	90,1
u250	20	14,3	6,6	0,32%	0,3	1,9	1.730,6	86,5	170,6
u500	20	11,0	11,9	0,30%	0,6	2,0	5.327,2	266,4	358,4
u1000	20	7,3	15,4	0,19%	0,8	3,0	19.318,1	965,9	1.083,0
t60	20	19,4	1,0	0,25%	0,1	1,0	305,2	15,3	63,1
t120	20	17,7	2,3	0,29%	0,1	1,0	963,5	48,2	109,2
t249	20	19,5	2,3	0,14%	0,1	1,7	1.643,6	82,2	138,1
t501	20	18,2	2,1	0,06%	0,1	1,3	5.331,6	266,6	370,7
DT1	720	637,3	96,2	0,12%	0,1	3,0	194.849,1	270,6	2.498,2
DT2	480	403,3	121,7	0,60%	0,3	7,5	85.344,6	177,8	7.178,7
Totais	1.360	1.159,2	268,3	0,22%	0,2	7,5	315.622,8	232,1	7.178,7

Fonte: Autor, 2014.

Tabela 23 – Resultados FF_VNS – Média de 10 testes.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	7,4	12,8	1,30%	0,6	1,2	661,9	33,1	54,8
u250	20	3,9	17,3	0,85%	0,9	1,7	4.070,4	203,5	311,9
u500	20	1,2	26,4	0,66%	1,3	3,5	25.541,6	1.277,1	1.938,3
u1000	20	0,0	55,9	0,70%	2,8	7,3	184.547,7	9.227,4	12.455,7
t60	20	0,0	20,1	5,03%	1,0	1,1	256,7	12,8	20,0
t120	20	0,0	21,9	2,74%	1,1	1,8	1.179,8	59,0	106,9
t249	20	0,0	27,1	1,63%	1,4	2,4	7.291,1	364,6	695,1
t501	20	0,0	48,1	1,44%	2,4	5,9	47.175,6	2.358,8	4.569,4
DT1	720	588,1	147,8	0,19%	0,2	2,5	24.966,5	34,7	347,7
DT2	480	379,3	145,6	0,72%	0,3	6,7	96.486,3	201,0	3.323,3
Totais	1.360	979,9	523,0	0,44%	0,4	7,3	392.177,6	288,4	12.455,7

Fonte: Autor, 2014.

Tabela 24 – Resultados FFD_VNS – Média de 10 testes.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	9,0	11,0	1,12%	0,6	1,0	165,3	8,3	22,2
u250	20	1,6	21,0	1,03%	1,1	2,0	617,9	30,9	56,9
u500	20	0,0	32,3	0,80%	1,6	2,9	2.064,1	103,2	187,1
u1000	20	0,0	54,2	0,68%	2,7	4,9	7.120,9	356,0	654,6
t60	20	0,0	20,0	5,00%	1,0	1,0	433,5	21,7	32,0
t120	20	0,0	20,3	2,54%	1,0	1,3	2.296,5	114,8	164,4
t249	20	0,0	24,5	1,48%	1,2	2,3	16.031,3	801,6	1.125,3
t501	20	0,0	40,4	1,21%	2,0	3,4	117.315,3	5.865,8	7.858,9
DT1	720	590,1	147,1	0,19%	0,2	2,7	24.810,7	34,5	319,6
DT2	480	380,1	144,7	0,71%	0,3	6,7	96.268,2	200,6	3.128,7
Totais	1.360	980,8	515,5	0,43%	0,4	6,7	267.123,7	196,4	7.858,9

Fonte: Autor, 2014.

Tabela 25 – Resultados BF_VNS – Média de 10 testes.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	7,9	12,3	1,25%	0,6	1,2	611,3	30,6	52,6
u250	20	3,4	17,9	0,88%	0,9	1,8	3.639,6	182,0	276,8
u500	20	1,0	26,8	0,67%	1,3	2,9	24.744,5	1.237,2	1.799,4
u1000	20	0,0	56,4	0,70%	2,8	6,1	166.476,0	8.323,8	12.069,3
t60	20	0,0	20,0	5,00%	1,0	1,0	268,4	13,4	24,5
t120	20	0,0	21,8	2,73%	1,1	1,7	1.219,3	61,0	103,1
t249	20	0,0	29,0	1,75%	1,5	3,2	7.189,6	359,5	666,2
t501	20	0,0	51,1	1,53%	2,6	6,1	46.128,6	2.306,4	4.575,6
DT1	720	586,8	149,8	0,19%	0,2	2,6	24.637,5	34,2	324,5
DT2	480	380,3	144,0	0,71%	0,3	6,9	97.589,7	203,3	3.349,4
Totais	1.360	979,4	529,1	0,44%	0,4	6,9	372.504,5	273,9	12.069,3

Fonte: Autor, 2014.

Tabela 26 – Resultados BFD_VNS – Média de 10 testes.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	8,7	11,3	1,15%	0,6	1,0	164,3	8,2	20,1
u250	20	1,4	20,8	1,02%	1,0	2,0	646,1	32,3	63,3
u500	20	0,0	32,5	0,81%	1,6	3,0	2.102,9	105,1	211,0
u1000	20	0,0	54,8	0,68%	2,7	5,6	7.137,7	356,9	663,8
t60	20	0,0	20,0	5,00%	1,0	1,0	427,2	21,4	30,3
t120	20	0,0	20,7	2,59%	1,0	1,4	2.288,9	114,4	169,3
t249	20	0,0	24,1	1,45%	1,2	2,2	16.348,8	817,4	1.114,7
t501	20	0,0	40,1	1,20%	2,0	3,6	118.176,0	5.908,8	7.461,7
DT1	720	585,4	149,6	0,19%	0,2	2,8	24.679,6	34,3	339,7
DT2	480	379,5	145,0	0,72%	0,3	6,7	97.653,0	203,4	3.084,1
Totais	1.360	975,0	518,9	0,43%	0,4	6,7	269.624,5	198,3	7.461,7

Fonte: Autor, 2014.

Tabela 27 – Resultados WF_VNS – Média de 10 testes.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	7,1	13,1	1,33%	0,7	1,2	1.096,1	54,8	83,5
u250	20	4,5	16,5	0,81%	0,8	1,6	7.430,3	371,5	564,0
u500	20	1,0	25,6	0,64%	1,3	2,6	51.969,7	2.598,5	3.709,0
u1000	20	0,0	53,8	0,67%	2,7	7,5	392.727,6	19.636,4	28.728,7
t60	20	0,0	20,0	5,00%	1,0	1,0	305,5	15,3	26,0
t120	20	0,0	21,3	2,66%	1,1	1,7	1.228,9	61,4	114,3
t249	20	0,0	28,9	1,74%	1,4	3,0	7.985,6	399,3	770,4
t501	20	0,0	47,0	1,41%	2,4	5,1	53.488,9	2.674,4	4.894,0
DT1	720	504,7	258,4	0,33%	0,4	3,6	88.805,8	123,3	1.868,8
DT2	480	368,0	156,5	0,77%	0,3	6,8	104.618,2	218,0	3.215,2
Totais	1.360	885,3	641,1	0,53%	0,5	7,5	709.656,6	521,8	28.728,7

Fonte: Autor, 2014.

Tabela 28 – Resultados WFD_VNS – Média de 10 testes.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	6,9	13,1	1,33%	0,7	1,0	352,1	17,6	37,7
u250	20	1,8	20,7	1,02%	1,0	2,0	1.337,6	66,9	134,6
u500	20	0,1	31,0	0,77%	1,6	2,7	5.255,5	262,8	540,0
u1000	20	0,0	52,9	0,66%	2,6	4,3	23.569,8	1.178,5	2.722,5
t60	20	0,0	20,0	5,00%	1,0	1,0	430,7	21,5	30,0
t120	20	0,0	20,4	2,55%	1,0	1,3	2.273,7	113,7	171,1
t249	20	0,0	24,7	1,49%	1,2	2,0	16.334,7	816,7	1.155,5
t501	20	0,0	40,9	1,22%	2,0	3,5	120.416,9	6.020,8	7.668,9
DT1	720	503,3	260,4	0,33%	0,4	4,0	88.010,3	122,2	1.901,9
DT2	480	366,6	157,0	0,78%	0,3	6,7	104.997,9	218,7	3.296,5
Totais	1.360	878,7	641,1	0,53%	0,5	6,7	362.979,2	266,9	7.668,9

Fonte: Autor, 2014.

Tabela 29 – Resultados MBS_VNS – Média de 10 testes.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	12,7	7,8	0,79%	0,4	1,4	370,7	18,5	31,3
u250	20	11,9	8,6	0,42%	0,4	1,5	1.358,6	67,9	91,4
u500	20	11,5	10,6	0,26%	0,5	2,0	4.868,2	243,4	286,5
u1000	20	7,0	15,7	0,20%	0,8	2,8	18.930,9	946,5	1.063,8
t60	20	0,0	20,0	5,00%	1,0	1,0	191,7	9,6	10,8
t120	20	0,0	20,0	2,50%	1,0	1,0	434,0	21,7	24,4
t249	20	0,0	20,3	1,22%	1,0	1,3	1.404,2	70,2	95,0
t501	20	0,0	24,8	0,74%	1,2	2,1	5.079,2	254,0	332,8
DT1	720	408,2	577,7	0,74%	0,8	8,5	93.817,4	130,3	1.446,7
DT2	480	416,3	74,5	0,37%	0,2	4,2	94.219,9	196,3	14.412,4
Totais	1.360	867,6	780,0	0,65%	0,6	8,5	220.674,8	162,3	14.412,4

Fonte: Autor, 2014.

Tabela 30 – Resultados MBS'_VNS – Média de 10 testes.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	12,8	7,2	0,73%	0,4	1,0	361,4	18,1	33,6
u250	20	14,0	6,3	0,31%	0,3	1,3	1.318,3	65,9	93,0
u500	20	11,6	10,0	0,25%	0,5	2,0	4.854,3	242,7	283,1
u1000	20	7,0	15,7	0,20%	0,8	3,0	18.664,0	933,2	1.016,5
t60	20	0,0	20,0	5,00%	1,0	1,0	201,0	10,1	12,6
t120	20	0,0	20,0	2,50%	1,0	1,0	460,4	23,0	30,7
t249	20	0,0	20,8	1,25%	1,0	1,6	1.610,8	80,5	121,4
t501	20	0,0	29,4	0,88%	1,5	2,3	5.654,1	282,7	427,2
DT1	720	639,8	87,7	0,11%	0,1	2,5	80.869,9	112,3	728,0
DT2	480	418,2	69,2	0,34%	0,1	3,8	62.136,8	129,5	7.185,5
Totais	1.360	1.103,4	286,3	0,24%	0,2	3,8	176.131,0	129,5	7.185,5

Fonte: Autor, 2014.

Tabela 31 – Resultados FF_WABP.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	18	2	0,20%	0,1	1	431	21,6	115
u250	20	17	3	0,15%	0,2	1	4.231	211,6	469
u500	20	12	9	0,22%	0,5	2	22.440	1.122,0	1.599
u1000	20	5	24	0,30%	1,2	4	108.490	5.424,5	6.095
t60	20	0	21	5,25%	1,1	2	403	20,2	22
t120	20	0	20	2,50%	1,0	1	1.374	68,7	80
t249	20	0	27	1,63%	1,4	3	5.797	289,9	317
t501	20	0	33	0,99%	1,7	3	22.633	1.131,7	1.242
DT1	720	665	55	0,07%	0,1	1	237.527	329,9	2.252
DT2	480	408	128	0,63%	0,3	9	97.411	202,9	2.782
Totais	1.360	1.125	322	0,27%	0,2	9	500.737	368,2	6.095

Fonte: Autor, 2014.

Tabela 32 – Resultados FFD_WABP.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	17	3	0,31%	0,2	1	358	17,9	113
u250	20	16	4	0,20%	0,2	1	3.009	150,5	392
u500	20	17	3	0,07%	0,2	1	7.915	395,8	1.418
u1000	20	17	3	0,04%	0,2	1	43.189	2.159,5	5.836
t60	20	0	21	5,25%	1,1	2	421	21,1	24
t120	20	0	27	3,38%	1,4	2	1.564	78,2	90
t249	20	0	60	3,61%	3,0	4	6.565	328,3	358
t501	20	0	114	3,41%	5,7	9	26.137	1.306,9	1.540
DT1	720	665	55	0,07%	0,1	1	242.256	336,5	2.632
DT2	480	408	128	0,63%	0,3	9	97.637	203,4	2.800
Totais	1.360	1.140	418	0,35%	0,3	9	429.051	315,5	5.836

Fonte: Autor, 2014.

Tabela 33 – Resultados BF_WABP.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	18	2	0,20%	0,1	1	425	21,3	110
u250	20	15	5	0,25%	0,3	1	4.674	233,7	411
u500	20	8	13	0,32%	0,7	2	25.253	1.262,7	1.818
u1000	20	6	17	0,21%	0,9	3	107.735	5.386,8	6.639
t60	20	0	20	5,00%	1,0	1	411	20,6	25
t120	20	0	21	2,63%	1,1	2	1.472	73,6	91
t249	20	0	26	1,57%	1,3	3	5.816	290,8	321
t501	20	0	32	0,96%	1,6	3	22.373	1.118,7	1.189
DT1	720	669	51	0,07%	0,1	1	238.364	331,1	2.240
DT2	480	406	130	0,64%	0,3	9	97.891	203,9	2.621
Totais	1.360	1.122	317	0,26%	0,2	9	504.414	370,9	6.639

Fonte: Autor, 2014.

Tabela 34 – Resultados BFD_WABP.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	17	3	0,31%	0,2	1	352	17,6	113
u250	20	16	4	0,20%	0,2	1	2.976	148,8	377
u500	20	17	3	0,07%	0,2	1	7.870	393,5	1.482
u1000	20	17	3	0,04%	0,2	1	43.207	2.160,4	5.730
t60	20	0	21	5,25%	1,1	2	505	25,3	34
t120	20	0	27	3,38%	1,4	2	1.830	91,5	117
t249	20	0	60	3,61%	3,0	4	6.549	327,5	357
t501	20	0	114	3,41%	5,7	9	25.433	1.271,7	1.362
DT1	720	669	51	0,07%	0,1	1	238.155	330,8	2.236
DT2	480	406	130	0,64%	0,3	9	97.104	202,3	2.605
Totais	1.360	1.142	416	0,35%	0,3	9	423.981	311,8	5.730

Fonte: Autor, 2014.

Tabela 35 – Resultados WF_WABP.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	17	3	0,31%	0,2	1	499	25,0	101
u250	20	16	4	0,20%	0,2	1	4.325	216,3	416
u500	20	16	4	0,10%	0,2	1	21.902	1.095,1	1.770
u1000	20	0	143	1,79%	7,2	16	124.111	6.205,6	6.417
t60	20	0	20	5,00%	1,0	1	426	21,3	40
t120	20	0	21	2,63%	1,1	2	1.392	69,6	80
t249	20	0	25	1,51%	1,3	2	5.824	291,2	326
t501	20	0	33	0,99%	1,7	4	22.763	1.138,2	1.223
DT1	720	634	92	0,12%	0,1	2	265.168	368,3	2.229
DT2	480	414	125	0,62%	0,3	9	94.787	197,5	2.644
Totais	1.360	1.097	470	0,39%	0,3	16	541.197	397,9	6.417

Fonte: Autor, 2014.

Tabela 36 – Resultados WFD_WABP.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	17	3	0,31%	0,2	1	516	25,8	100
u250	20	16	4	0,20%	0,2	1	3.518	175,9	404
u500	20	13	7	0,17%	0,4	1	16.954	847,7	1.601
u1000	20	11	10	0,12%	0,5	2	83.768	4.188,4	6.148
t60	20	0	24	6,00%	1,2	2	425	21,3	25
t120	20	0	27	3,38%	1,4	2	1.568	78,4	92
t249	20	0	60	3,61%	3,0	4	6.520	326,0	352
t501	20	0	120	3,59%	6,0	8	25.633	1.281,7	1.347
DT1	720	634	92	0,12%	0,1	2	266.129	369,6	2.234
DT2	480	414	125	0,62%	0,3	9	94.956	197,8	2.657
Totais	1.360	1.105	472	0,39%	0,3	9	499.987	367,6	6.148

Fonte: Autor, 2014.

Tabela 37 – Resultados MBS_WABP.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	17	3	0,31%	0,2	1	607	30,4	130
u250	20	18	2	0,10%	0,1	1	2.993	149,7	486
u500	20	18	2	0,05%	0,1	1	8.170	408,5	1.611
u1000	20	17	3	0,04%	0,2	1	33.509	1.675,5	6.193
t60	20	0	20	5,00%	1,0	1	481	24,1	28
t120	20	0	20	2,50%	1,0	1	1.656	82,8	94
t249	20	0	20	1,20%	1,0	1	6.154	307,7	366
t501	20	0	20	0,60%	1,0	1	23.604	1.180,2	1.342
DT1	720	405	499	0,64%	0,7	6	313.523	435,4	2.579
DT2	480	463	17	0,08%	0,0	1	125.285	261,0	14.170
Totais	1.360	938	606	0,51%	0,4	6	515.982	379,4	14.170

Fonte: Autor, 2014.

Tabela 38 – Resultados MBS'_WABP.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	19	1	0,10%	0,1	0	391	19,6	103
u250	20	19	1	0,05%	0,1	1	2.684	134,2	419
u500	20	19	1	0,02%	0,1	1	6.669	333,5	1.643
u1000	20	18	2	0,02%	0,1	1	33.711	1.685,6	6.212
t60	20	0	20	5,00%	1,0	1	469	23,5	27
t120	20	0	20	2,50%	1,0	1	1.605	80,3	103
t249	20	0	21	1,27%	1,1	2	6.484	324,2	383
t501	20	0	22	0,66%	1,1	2	23.749	1.187,5	1.254
DT1	720	680	40	0,05%	0,1	1	289.991	402,8	2.655
DT2	480	464	16	0,08%	0,0	1	93.847	195,5	7.212
Totais	1.360	1.219	144	0,12%	0,1	2	459.600	337,9	7.212

Fonte: Autor, 2014.

Tabela 39 – Resultados AugNN-FFD – Média de 10 testes.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	12,5	7,5	0,76%	0,4	1,0	2.802,7	140,1	328,1
u250	20	7,9	12,1	0,59%	0,6	1,0	18.947,9	947,4	1.237,1
u500	20	0,0	26,2	0,65%	1,3	2,0	85.530,1	4.276,5	4.453,5
u1000	20	0,0	57,7	0,72%	2,9	4,5	329.607,6	16.480,4	16.791,4
t60	20	0,0	21,0	5,25%	1,1	1,7	1.837,0	91,9	97,9
t120	20	0,0	57,3	7,16%	2,9	3,1	5.885,5	294,3	316,3
t249	20	0,0	146,7	8,84%	7,3	8,0	22.070,7	1.103,5	1.175,3
t501	20	0,0	323,8	9,69%	16,2	17,0	83.039,7	4.152,0	4.413,9
DT1	720	619,7	132,0	0,17%	0,2	3,0	836.946,9	1.162,4	5.129,8
DT2	480	332,6	397,4	1,96%	0,8	12,9	286.998,0	597,9	4.306,3
Totais	1.360	972,7	1.181,7	0,99%	0,9	17,0	1.673.666,1	1.230,6	16.791,4

Fonte: Autor, 2014.

Tabela 40 – Resultados AugNN-BFD – Média de 10 testes.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	10,0	10,0	1,02%	0,5	1,0	6.021,7	301,1	650,6
u250	20	4,1	16,9	0,83%	0,8	2,0	40.208,0	2.010,4	2.464,7
u500	20	3,3	16,8	0,42%	0,8	1,1	163.559,3	8.178,0	8.698,1
u1000	20	0,0	38,5	0,48%	1,9	3,1	649.794,9	32.489,7	32.918,9
t60	20	0,0	22,8	5,70%	1,1	2,0	3.758,0	187,9	322,0
t120	20	0,0	45,8	5,73%	2,3	3,0	11.427,3	571,4	608,7
t249	20	0,0	118,0	7,11%	5,9	7,1	42.598,7	2.129,9	2.266,6
t501	20	0,0	266,6	7,98%	13,3	15,0	159.963,5	7.998,2	8.312,1
DT1	720	598,4	142,1	0,18%	0,2	2,7	1.674.825,2	2.326,1	9.868,2
DT2	480	343,3	384,5	1,90%	0,8	13,6	543.764,2	1.132,8	8.245,3
Totais	1.360	959,1	1.062,0	0,89%	0,8	15,0	3.295.920,8	2.423,5	32.918,9

Fonte: Autor, 2014.

Tabela 41 – Resultados AugNN-WFD – Média de 10 testes.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	3,9	18,9	1,92%	0,9	2,0	11.588,2	579,4	1.007,5
u250	20	0,0	29,4	1,45%	1,5	3,0	65.519,4	3.276,0	3.787,9
u500	20	0,0	49,4	1,23%	2,5	4,0	258.291,5	12.914,6	13.507,2
u1000	20	0,0	98,8	1,23%	4,9	8,0	1.015.577,0	50.778,9	51.328,5
t60	20	0,0	26,4	6,60%	1,3	2,9	5.688,6	284,4	420,7
t120	20	0,0	40,7	5,09%	2,0	2,8	17.611,4	880,6	928,3
t249	20	0,0	99,8	6,01%	5,0	6,4	65.603,1	3.280,2	3.493,4
t501	20	0,0	246,6	7,38%	12,3	13,8	246.423,5	12.321,2	12.642,1
DT1	720	464,5	383,9	0,49%	0,5	4,8	2.758.520,3	3.831,3	15.292,6
DT2	480	297,6	459,1	2,27%	1,0	13,0	902.339,3	1.879,9	12.764,0
Totais	1.360	766,0	1.453,0	1,21%	1,1	13,8	5.347.162,3	3.931,7	51.328,5

Fonte: Autor, 2014.

Tabela 42 – Resultados método híbrido *Perturbation* MBS' + VNS – Média de 10 testes.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	13,4	6,6	0,67%	0,3	1,0	913,8	45,7	109,4
u250	20	14,6	5,5	0,27%	0,3	1,1	1.850,1	92,5	196,8
u500	20	11,7	10,0	0,25%	0,5	2,0	5.497,0	274,9	403,6
u1000	20	8,3	12,8	0,16%	0,6	2,1	19.383,3	969,2	1.154,0
t60	20	19,8	0,2	0,05%	0,0	0,2	333,8	16,7	60,6
t120	20	19,0	1,0	0,13%	0,1	0,6	889,9	44,5	106,7
t249	20	18,8	1,2	0,07%	0,1	0,6	1.720,3	86,0	151,7
t501	20	17,9	2,3	0,07%	0,1	1,0	5.379,2	269,0	398,2
DT1	720	643,2	83,0	0,11%	0,1	2,2	230.815,9	320,6	3.278,8
DT2	480	433,4	53,1	0,26%	0,1	3,0	89.844,5	187,2	7.138,1
Totais	1.360	1.200,1	175,7	0,15%	0,1	3,0	356.627,8	262,2	7.138,1

Fonte: Autor, 2014.

Tabela 43 – Resultados método híbrido MBS'_Pert_WABP – Média de 10 testes.

	Problemas	Sucessos (problemas)	Erro total (objetos)	Erro total %	Erro médio (objetos)	Erro máximo (objetos)	Tempo total (ms)	Tempo médio (ms)	Tempo máximo (ms)
u120	20	18,6	1,6	0,16%	0,1	0,9	1.015,1	50,8	168,1
u250	20	18,3	1,7	0,08%	0,1	1,0	3.422,5	171,1	545,9
u500	20	17,5	2,5	0,06%	0,1	1,0	8.962,0	448,1	1.688,9
u1000	20	18,5	1,7	0,02%	0,1	1,0	29.809,3	1.490,5	5.870,8
t60	20	19,4	0,7	0,17%	0,0	0,4	343,1	17,2	69,9
t120	20	19,3	0,9	0,11%	0,0	0,5	933,3	46,7	124,2
t249	20	19,0	1,1	0,07%	0,1	0,7	1.958,2	97,9	314,6
t501	20	18,0	2,0	0,06%	0,1	1,0	7.283,5	364,2	1.355,5
DT1	720	674,5	45,6	0,06%	0,1	1,1	448.318,9	622,7	5.131,6
DT2	480	472,4	7,6	0,04%	0,0	1,0	108.741,3	226,5	7.172,6
Totais	1.360	1.295,5	65,3	0,05%	0,0	1,1	610.787,2	449,1	7.172,6

Fonte: Autor, 2014.