

UNIVERSIDADE FEDERAL DE UBERLÂNDIA – UFU
FACULDADE DE CIÊNCIAS INTEGRADAS DO PONTAL
– FACIP GRADUAÇÃO EM ENGENHARIA DE
PRODUÇÃO

BEATRIZ BONATO TORQUATO

O PROBLEMA DE AGRUPAMENTO DE TAREFAS EM UMA
FROTA HETEROGÊNEA DE VEÍCULOS

ITUIUTABA
MARÇO 2017

BEATRIZ BONATO TORQUATO

**O PROBLEMA DE AGRUPAMENTO DE TAREFAS EM UMA
FROTA HETEROGÊNEA DE VEÍCULOS**

Trabalho de conclusão de curso exigido como requisito obrigatório para a obtenção do título de bacharel em Engenharia de Produção da Universidade Federal de Uberlândia.

Área de Concentração: Pesquisa Operacional.

Orientador: Jorge von Atzingen dos Reis.

ITUIUTABA

MARÇO 2017

O problema do agrupamento de tarefas em uma frota heterogênea de veículos

Trabalho de conclusão de curso
aprovado para obtenção do título de
bacharel em Engenharia de
Produção da Universidade Federal
de Uberlândia pela banca
examinadora formada por:

Banca de Avaliação:

Prof. Dr. Jorge von Atzingen dos Reis

UFU Orientador

Prof. Dr. Antonio Álvaro de Assis Moura

UFU Membro

Prof. Dr. Fernando de Araújo

UFU Membro

Ituiutaba, Março de 2017.

DEDICATÓRIA

Dedico este trabalho em primeiro lugar a Deus por ter me dado saúde e força para superar às dificuldades, aos meus pais e meus irmãos, pelo amor, incentivo e apoio incondicional e a todos meus amigos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

AGRADECIMENTOS

Agradeço à Instituição pelo ambiente criativo e amigável que proporciona, ao meu orientador Jorge Reis pelo empenho dedicado à elaboração deste trabalho, à UFU e PROPP/CNPq pelo financiamento da bolsa, e a todo corpo docente por compartilhar sua experiência e ensinamentos durante esses anos de graduação, em especial aos professores presentes nesta banca Antonio Álvaro Assis Moura e Fernando de Araújo.

RESUMO

Diante da globalização e grande competição no mercado, as empresas começaram a buscar novas alternativas competitivas, assim qualquer ganho com economia de custo passou a ser importante, desta forma surgiu o planejamento da distribuição das mercadorias em diversos tipos de modais. O custo do transporte de produtos é cerca de 60% do custo total de toda a logística, de 4% a 25% do faturamento bruto da empresa (RIBEIRO & FERREIRA, 2002). Desta forma, determinar o número de veículos necessários para o transporte de uma determinada quantidade de produtos é um dos principais problemas da área de logística. Neste trabalho será abordado o problema de agrupamento de entregas em uma frota heterogênea de veículos (*Variable Size Bin Packing Problem* – VSBPP). O VSBPP consiste em, dado um conjunto de objetos (entregas), determinar qual o número mínimo de *bins* (veículos) necessários para transportar todos os objetos.

Palavras-chaves: *Bin Packing Problem*, Meta-heurística, Busca em Vizinhança Variável.

ABSTRACT

In the face of globalization and high competition in the market, companies began to seek new competitive alternatives, so any gains from cost savings became important, thus came the planning of the distribution of goods in various types of modes. The cost of transporting products is about 60% of the total cost of all the logistics, from 4% to 25% of gross sales of the company (RIBEIRO & FERREIRA, 2002). Thus, determining the number of vehicles needed to transport a given amount of product is one of the main problems of logistics. This work will address the delivery clustering problem in a heterogeneous fleet of vehicles (Variable Size Bin Packing Problem - VSBPP). The VSBPP consists of, given a set of objects (deliveries), which determine the minimum number of bins (vehicle) necessary to move all the objects.

Keywords: Bin Packing Problem, Meta-heuristic, Variable Neighborhood Search.

SUMÁRIO

1. INTRODUÇÃO.....	1
2. REFERENCIAL TEÓRICO.....	4
2.1. Pesquisa operacional e Otimização.....	4
2.2. O <i>Bin Packing Problem</i>	5
2.3. Heurística e Meta-heurística.....	6
2.3.1. Heurística.....	6
2.3.2. <i>Meta-heurística</i>	7
2.4 Busca em Vizinhança Variável (<i>Variable Neighborhood Search</i> – VNS).....	8
2.5 Correlação Estatística.....	8
2.6 <i>First-Fit decreasing algorithm</i>	9
3. METODOLOGIA.....	11
3.1. Leitura dos dados e Saída dos dados.....	11
3.2. O algoritmo.....	12
3.3 Exemplo.....	14
4. ANÁLISE DOS RESULTADOS OBTIDOS.....	16
5. CONSIDERAÇÕES FINAIS.....	23
6. REFERÊNCIAS.....	25
ANEXO A – Código uma meta-heurística VNS aplicada ao problema VSBPP.....	28
ANEXO B – Implementação do GUROBI para o problema VSBPP.....	45

LISTA DE TABELAS

<u>Tabela 1: Instância Bin Packing Problem 1.....</u>	<u>15</u>
<u>Tabela 2: Restrições e Resultados Esperados.....</u>	<u>15</u>
<u>Tabela 3: Exemplo de formação de frotas.....</u>	<u>15</u>
<u>Tabela 4: Representação de formação de frotas.....</u>	<u>16</u>
<u>Tabela 5: Carregamento completo da frota para o problema 1.....</u>	<u>16</u>
<u>Tabela 6: Resultados obtidos para os 20 problemas da instância 1.....</u>	<u>18</u>
<u>Tabela 7: Resultados obtidos utilizando o Input Analyser, os problemas em amarelo são os que ultrapassarem as três horas no tempo de processamento.....</u>	<u>20</u>
<u>Tabela 8: Matriz de correlações de Pearson para Instância 1, matriz obtida através do software Rstudio©.....</u>	<u>21</u>
<u>Tabela 9: Tempos computacionais obtidos nos testes utilizando LINGO©.....</u>	<u>22</u>
<u>Tabela 10: Tempos computacionais obtidos nos testes utilizando GUROBI©.....</u>	<u>22</u>

LISTA DE FIGURAS

<u>Figura 1: Custos logísticos das empresas no Brasil em relação à receita líquida em 2012, por setor.....</u>	<u>2</u>
<u>Figura 2: Modelo matemático do <i>Bin Packing Problem</i>.....</u>	<u>5</u>
<u>Figura 3: Exemplo de algoritmo <i>First Fit Decreasing</i>.....</u>	<u>10</u>
<u>Figura 4: Exemplo de algoritmo <i>First Fit</i>.....</u>	<u>10</u>
<u>Figura 5: Etapas do desenvolvimento do trabalho.....</u>	<u>11</u>
<u>Figura 6: Pseudocódigo Meta-heurística Busca em Vizinhaça Variável.....</u>	<u>13</u>
<u>Figura 7: Funcionamento do VNS.....</u>	<u>14</u>
<u>Figura 8: Gráfico Tempo vs Número de Objetos.....</u>	<u>17</u>

1. INTRODUÇÃO

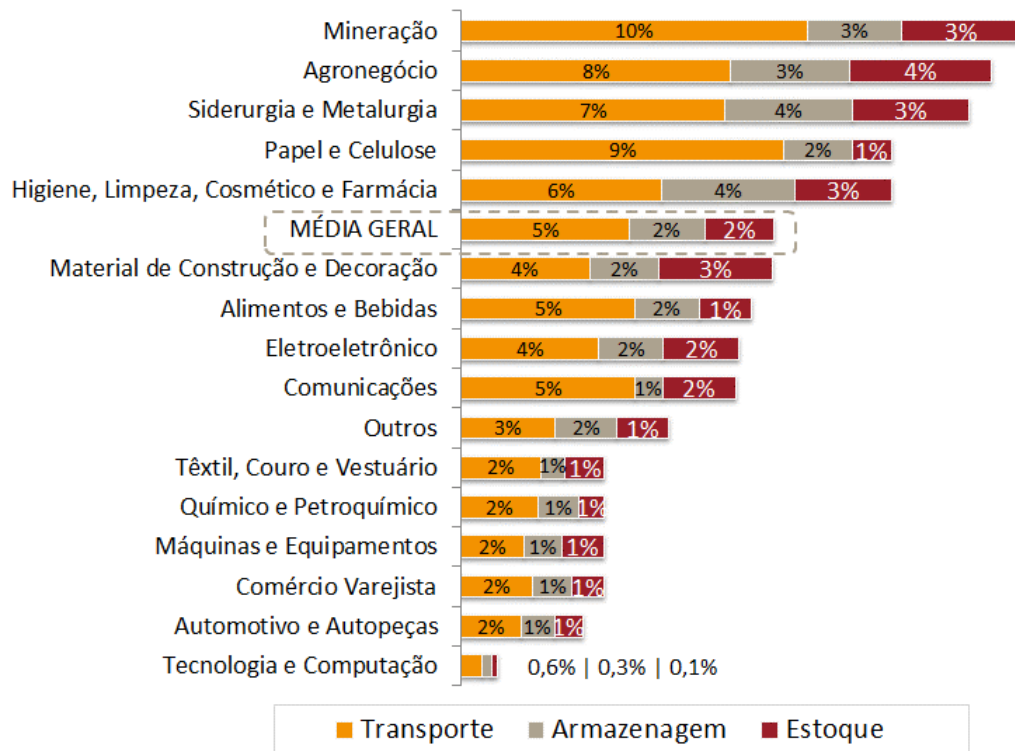
Atualmente, no setor industrial, a competitividade do mercado é a motivação para buscar melhorias em todos os departamentos da empresa. Essas melhorias procuram diminuir o custo e o tempo de produção, a fim de satisfazer o cliente.

Segundo SANTOS & BASSANESI (2006) entre os anos de 1.910 e 1.970 as estratégias operacionais de uma empresa eram focadas prioritariamente na produção, pois o mercado possuía uma imensa demanda e quanto maior fosse a oferta, maior seria o lucro. Com o surgimento do Toyotismo essa realidade mudou, enquanto o Fordismo priorizava a oferta, o Toyotismo possuía estratégia na demanda, visando satisfazer as preferências do consumidor e desta forma a logística se tornou algo importante para o mercado. Uma vez que estes clientes exigentes, optavam por produtos de maior qualidade com o menor preço, escolhendo assim, as empresas que permanecerão no mercado.

O crescente número de itens com diferentes padrões de demanda e características específicas aumentaram a complexidade da administração de materiais devido à necessidade de controle diferenciado (SANTOS & RODRIGUES, 2006). Com intuito de solucionar problemas e alcançar objetivos, as empresas buscam nas estratégias operacionais uma solução para atender melhor os seus clientes. Desta forma o planejamento estratégico logístico se tornou algo fundamental para o mercado competitivo do cenário atual.

De acordo com RIBEIRO & FERREIRA (2002), 4% a 25% do faturamento bruto de uma empresa é composta por custos logísticos, dentre as principais áreas da logística listadas por ROSA (2007) sendo elas: os estoques, processamento de pedidos e o transporte. Pode-se dizer que este último é mais importante, pois RIBEIRO & FERREIRA (2002), ainda afirmam que o custo do transporte de produtos é cerca de 60% do custo total de toda a logística. Esses custos de transportes que são considerados uns dos maiores custos logísticos e tem grande relevância na formação do preço final do produto, ele é composto por todas as despesas realizadas na movimentação de determinado produto desde a origem até seu destino final. LIMA (2014) utiliza a Figura 1 para ilustrar melhor esses dados estatísticos, onde as porcentagens nas barras representam os custos logísticos em relação a receita da empresa, e a legenda de cores dividi o custo logístico em custos com transporte, armazenagem e estoques.

Figura 1: Custos logísticos das empresas no Brasil em relação à receita líquida em 2012, por setor



Fonte: ILOS (2012)

Entretanto, os problemas relacionados à área de transporte não envolvem apenas escolher o tipo de modal para a alocação dos produtos. Para FLEURY (2002) uma dimensão a ser considerada na escolha do modal é a qualidade dos serviços oferecidos, são cinco as dimensões mais importantes, no que diz respeito às características dos serviços oferecidos: velocidade; consistência; capacitação; disponibilidade; e frequência. RIBEIRO & FERREIRA (2002), desenvolvem este conceito e citam os elementos a seguir como os principais: “... peso e volume, densidade média; dimensão da carga; dimensão do veículo; grau de fragilidade da carga; grau de perecibilidade; estado físico; assimetria; e compatibilidade entre cargas diversas”.

Na literatura estes critérios que devem ser avaliados no planejamento logístico são conhecidos o Problema de *Bin Packing* e suas variantes. Como tema central este trabalho abordará a variante *Variable Size Bin Packing Problem* – VSBPP. Esta variante VSBPP consiste em, dado um conjunto de objetos (encomendas), determinar qual o número mínimo de *bins* (neste caso são os veículos), com tamanhos variados, necessários para transportar todos os objetos (REIS & CUNHA, 2007).

Estes problemas possuem alto nível de complexidade e são classificados como NP-difícil, que são problemas determinísticos em tempo não polinomial. Em geral, são problemas que ao utilizar os métodos exatos, não se consegue encontrar soluções viáveis em tempo polinomial para problemas de grande proporção.

O VSBPP contém o BPP unidimensional clássico como um caso especial. No BPP todos os compartimentos têm a mesma capacidade e custo ($n = 1$). O BPP é um problema NP-hard e, portanto, o VSBPP é NP-hard também (CORREIA *et al*, 2008).

Como estes problemas são de extrema complexidade computacional, recorre-se a utilização de métodos mais eficientes para encontrar uma solução viável. Os métodos mais utilizados para a realização dos problemas são os métodos heurísticos e os métodos meta-heurísticos.

Os métodos heurísticos visam encontrar uma solução, que não seja necessariamente a melhor, mas seja a que possui um tempo computacional aceitável. Por outro lado as meta-heurísticas, que também resolvem problemas da classe NP-hard, são procedimentos destinados a encontrar uma boa solução, eventualmente ótima, consistindo na aplicação, em cada passo, de uma heurística subordinada, a qual tem que ser modelada para cada problema específico (SOUZA, 2008).

Através dessa técnica define-se o critério de otimização na qual envolve a minimização do número de veículos. Como comparação de resultado e comprovação de tempo computacional inviável foi utilizado o pacote de otimização GUROBI© e o pacote de otimização LINGO©.

Portanto, devido a escassez de estudos relacionados a esta variação do *Bin Packing Problem* este trabalho abordará a busca de soluções factíveis para 8 instâncias do problema do *Bin Packing Problem* cada uma composta por 20 problemas que variam em número de *bins*. Utilizando a meta-heurística a Busca em Vizinhaça Variável (*Variable Neighborhood Search*—VNS), GUROBI©, LINGO©, Input Analyser do Arena©, o software estatístico Rstudio© para resolver o VSBPP.

2. REFERENCIAL TEÓRICO

Nesta seção são apresentadas a revisão bibliográfica sobre pesquisa operacional e otimização, *Bin Packing Problem*, heurística e meta-heurística, Busca em Vizinhaça Variável – VNS, e Correlação Estatística.

2.1. PESQUISA OPERACIONAL E OTIMIZAÇÃO

“A pesquisa operacional é o enfoque científico sobre a tomada de decisões” (ARENALES *et al.* 2006, p.3). Os autores ainda relatam sobre o surgimento do termo que vem desde a invenção do radar em 1.934, sendo moldado na Segunda Guerra Mundial. Porém, foi ao final da guerra que a pesquisa operacional evolui mais rapidamente, e hoje tem sido chamada de ciência e tecnologia da decisão.

HILLIER e LIEBERMAN (2010) definem a pesquisa operacional em linhas gerais, como sendo “a descrição de um sistema organizado com o auxílio de um modelo, e através da experimentação com o modelo, na descoberta da melhor maneira de operar o sistema”.

O termo modelo utilizado na definição de HILLIER e LIEBERMAN (2010), é usado com um problema teórico, que possui as características principais do problema real que está sendo analisado. O modelo deixa de ser apenas um modelo para se tornar um modelo matemático quando passa a ser descrito por relações matemáticas. De acordo com ARENALES *et al.* (2006), o modelo matemático é uma representação abstrata do problema real.

Como toda pesquisa tem um objetivo, a pesquisa operacional não seria diferente. Na pesquisa operacional objetiva-se minimizar ou maximizar uma função através da atribuição de valores as variáveis reais ou inteiras, respeitando sempre as restrições que limitaram um conjunto de soluções factíveis de um determinado problema.

Neste trabalho o que irá ser buscado é uma solução ótima, ou seja, dentro das soluções possíveis a melhor solução que otimiza o problema de agrupamento de veículos numa frota heterogênea de veículos. A busca pela solução ótima se torna inviável à medida que o número de *bins* aumenta, pois a complexidade dos problemas vai se alterando. O que comprova a natureza NP-hard do VSBPP, “Uma vez que o VSBPP é NP-hard, se reduz ao BPP no caso particular em que todos os *bins* são idênticos, que é NP-hard” (GAREY &

JOHNSON, 1979). Por isso será aceito o uso dos modelos heurísticos e meta-heurísticos que objetivam encontrar uma solução próxima da ótima, também conhecida como solução estrela.

2.2. O BIN PACKING PROBLEM

O VSBPP é um problema de otimização combinatória de minimização de espaço, tempo e valor. Consiste no empacotamento de objetos de diferentes tamanhos ou pesos no menor número possível de *bins* heterogêneos (ALVIN, 2003). Enquanto no BPP os *bins* são uniformes, o VSBPP trabalha com *bins* de capacidades diferentes (REIS & CUNHA, 2007). Por isso se entende que o VSBPP é uma variante do *Bin Packing Problem* – BPP.

Na sua formulação matemática são utilizadas duas variáveis binárias: y_i e x_{ij} , onde essas caracterizam respectivamente a utilização do *bin* i e a alocação do item j no *bin* i . Caso os valores das variáveis seja igual a um representa que o *bin* i está sendo utilizado e o item j foi alocado ao *bin* i , e as variáveis recebem zero caso contrário (REIS & CUNHA, 2007).

A Figura 2 apresenta uma modelagem matemática retirada e adaptada de REIS & CUNHA (2007). A função objetivo (1) visa à minimização dos custos envolvidos na utilização dos *bins*.

Figura 2: Modelo matemático do Bin Packing Problem

$$\begin{aligned} \min \sum_{i=1}^m f_i y_i & (1) \\ \sum_{j=1}^n w_j x_{ij} & \leq b_i y_i, \forall i=1 \dots m & (2) \\ \sum_{i=1}^m x_{ij} & = 1, \forall j=1 \dots n & (3) \\ x_{ij} & \in \{0,1\} \forall i=1 \dots m \forall j=1 \dots n & (4) \\ y_i & \in \{0,1\} \forall i=1 \dots m & (5) \end{aligned}$$

Fonte: REIS e CUNHA (2007)

A restrição (2) impede que a capacidade do *bin* i seja violada, ou seja, o somatório de todos os itens que estão acomodados no *bin* i deve ser igual ou inferior ao peso do *bin*. A outra restrição (3) pressupõe que a soma do item j em todos os *bins* i deve ser igual a um, assim não é possível que parte de um item vai para um veículo enquanto a outra parte vai para

outro. As restrições 4 e 5 garantem que todas as variáveis sejam binárias e inteiras.

2.3. HEURÍSTICA E META-HEURÍSTICA

A resolução de problemas de otimização combinatória pode ser realizada utilizando algoritmos e técnicas, conhecidas como heurísticas e meta-heurísticas.

2.3.1. Heurística

“Define-se heurística como sendo uma técnica inspirada em processos intuitivos que procura uma boa solução a um custo computacional aceitável, sem, no entanto, estar capacitada a garantir sua otimização, bem como garantir quão próximo está da solução ótima.” (SOUZA, 2008, p.2).

As heurísticas são muito estudadas e utilizadas na prática, em geral, pois é suficiente para encontrar uma solução boa e aplicável para o problema reduzindo o esforço computacional consideravelmente procurando uma solução ótima. O objetivo é encontrar uma solução de boa qualidade em um tempo computacional reduzido e preferencialmente próxima da solução ótima, quando a solução ótima for conhecida.

Para DIAZ (1996), a heurística é um algoritmo que encontra uma solução factível e em um tempo razoável, mas essa solução não é necessariamente a solução ótima para um problema. Mas mesmo assim, existem vários casos em que se é vantajoso usar uma heurística para resolução de determinado problema, como situações em que o processo de resolução exato não está disponível, ou quando não é viável usar uma meta-heurística, pois nela o nível de dificuldade é maior e o custo de se gerar uma solução ótima também. Compreendendo que apenas uma solução factível já está de bom tamanho, visto que a solução ótima não gerará um ganho técnico ou econômico relevante.

A técnica heurística pode ser dividida em heurística construtiva e heurística de refinamento. A heurística construtiva é utilizada para gerar a solução inicial elemento por elemento, a qual depende do problema e do tipo de avaliação adotado para determinar a escolha de cada elemento. Se for utilizado a heurística construtiva gulosa somente o melhor elemento é inserido, e para saber qual é o melhor ocorre a análise de benefício de cada elemento. De acordo com (SOUZA, 2008) a grande vantagem deste método é que ele é de

fácil implementação. A desvantagem é baixa qualidade da solução final produzida, o que requererá um esforço maior na fase de refinamento.

As heurísticas construtivas são um tipo de heurística que constroem uma solução a partir de regras baseadas nos dados do problema. É adicionado em cada etapa um elemento da solução, como por exemplo, um dado, ou o valor de uma variável. Esse tipo de heurística pode gerar uma solução factível ou não. Destaca-se entre elas, a forma gulosa onde o elemento melhor classificado, de acordo com a função de avaliação, é selecionado. As soluções geradas por um algoritmo de construção gulosa geralmente são melhores que as obtidas de forma aleatória. (DORIGO e STUTZLE, 2003).

Na heurística de refinamento de acordo com SOUZA (2008) esta classe de heurística parte de uma solução inicial qualquer (a qual pode ser obtida por uma heurística construtiva ou então gerada aleatoriamente) e caminha, a cada iteração, de vizinho para vizinho de acordo com a definição de vizinhança adotada. A definição de vizinhança é crucial em uma heurística de refinamento. De uma solução s do espaço de soluções deve ser sempre possível atingir qualquer outra solução em um número finito de passos, utilizando um determinado tipo ou tipos de movimentos.

2.3.2. Meta-heurística

Segundo SOUZA (2008) uma meta-heurística é um conjunto de conceitos que pode ser utilizado para definir métodos heurísticos aplicáveis a um extenso conjunto de diferentes problemas. Em outras palavras, uma meta-heurística pode ser vista como uma estrutura algorítmica geral que pode ser aplicada a diferentes problemas de otimização com relativamente poucas modificações que possam adaptá-la a um problema específico.

É uma técnica que possui procedimentos para achar uma boa solução ou até uma solução ótima. Ocorre a aplicação de uma heurística subordinada, usada para determinado problema específico. Ainda de acordo com SOUZA (2008) as meta-heurísticas são diferenciadas de acordo com o método usado para fugir de ótimos locais ainda que distantes dos ótimos globais.

Neste trabalho as meta-heurísticas serão divididas em duas categorias: busca local e busca populacional. Essa divisão é de acordo com o método usado para explorar o espaço de soluções: busca local, onde essa exploração é realizada através de movimentos, gerando outra

solução promissora em sua vizinhança, os quais são realizados a cada passo sobre a solução corrente e a busca populacional: onde existe a tentativa de produzir soluções ainda melhores do que as encontradas e, para isso, precisa manter um conjunto de boas soluções e combiná-las para melhorar.

Uma possível divisão para as meta-heurísticas segundo MELIÁN *et al.* (2003) seria dividi-las em meta-heurística construtiva, meta-heurística de relaxação, meta-heurística de busca por entornos, meta-heurística evolutiva, meta-heurísticas populacionais e outros tipos que não se enquadram adequadamente em nenhuma das outras categorias citadas. De acordo com esta possível divisão, neste estudo utilizaremos uma meta-heurística de busca por entornos, a Busca em Vizinhança Variável (VNS).

2.4 BUSCA EM VIZINHANÇA VARIÁVEL (*VARIABLE NEIGHBORHOOD SEARCH* – VNS)

As meta-heurísticas são utilizadas para guiar as heurísticas de busca local, a fim de escapar de ótimos locais, ampliando a exploração do espaço de soluções viáveis. SOUZA (2008) diz que a Busca em Vizinhança Variável (*Variable Neighborhood Search* - VNS) se enquadra nesta categoria de heurísticas onde a busca local consiste em explorar o espaço de soluções por meio de trocas sistemáticas de estruturas de vizinhança, o VNS não segue uma trajetória, mas sim explora vizinhanças gradativamente mais “distantes” da solução corrente e focaliza a busca em torno de uma nova solução se e somente se um movimento de melhora é realizado.

Desta maneira, implementou-se neste trabalho a meta-heurística VNS para resolver o VSBPP, pois esta meta-heurística possui um procedimento de busca local a ser aplicado sobre a solução corrente, que se acreditou ser o mais eficaz para o problema que está sendo analisado.

2.5 CORRELAÇÃO ESTATÍSTICA

A correlação mensura a direção e o grau da relação linear entre duas variáveis quantitativas (MOORE, 2007). Ou seja, o coeficiente de correlação, na teoria de probabilidade e estatística, indica a força e a direção do relacionamento linear entre duas variáveis aleatórias. Neste estudo, o conceito foi utilizado para testar a correlação entre as

variáveis de entrada, que a princípio parecem similares mas apresentam resultados divergentes.

A correlação em geral se refere à medida de relação entre as variáveis, porém não explica a causalidade, que seria a relação entre um evento A e um evento B. No entanto, saber se existe alguma correlação entre as variáveis já seria o suficiente para tentar entender a origem dos diferentes resultados obtidos.

Ainda segundo MOORE (2007), o coeficiente de correlação entre produto e momento, mais conhecido como coeficiente de Pearson, mede o grau da correlação entre duas variáveis. Este coeficiente, normalmente representado por ρ assume apenas valores entre -1 e 1, o sinal indica direção positiva ou negativa do relacionamento e o valor sugere a força da relação entre as variáveis.

- $\rho=1$ significa uma correlação perfeita positiva entre as duas variáveis.;
- $\rho=-1$ significa uma correlação negativa perfeita entre as duas variáveis.

Isto é se uma aumenta, a outra sempre diminui.;

- $\rho=0$ significa que as duas variáveis não dependem linearmente uma da outra. No entanto, pode existir uma correlação não linear entre elas.

2.6 FIRST-FIT DECREASING ALGORITHM

De acordo com ANDERSON *et al.* (1989), existe uma abordagem ao se estudar sequencialmente algoritmos de aproximação para o problema do *Bin Packing*. Esta consiste em olhar para a simples heurística do problema e analisar seu comportamento. Um importante exemplo de tal heurística é o algoritmo *First Fit Decreasing* (FFD), ele considera os itens em ordem decrescente, e coloca cada item para o primeiro *bin* disponível que tenha o suficiente espaço restante. (Figura 3)

Figura 3: Exemplo de algoritmo *First Fit Decreasing*

Algoritmo <i>First Fit Decreasing</i>									
Exemplo: Alocar itens abaixo em bins de cap 6									
Itens:	4	1	2	5	3	2	3	6	3
Itens dec:	6	5	4	3	3	3	2	2	1

BIN 1	6									6
BIN 2	5	1								6
BIN 3	4	2								6
BIN 4	3	3								6
BIN 5	3	2								5
BIN 6										0

Fonte: Autoria Própria

Ao passo que, o algoritmo *First Fit*, apenas aloca de acordo com a distribuição inicial dos itens. (Figura 4)

Figura 4: Exemplo de algoritmo *First Fit*

Algoritmo <i>First Fit</i>									
Exemplo: Alocar itens abaixo em bins de cap 6									
Itens:	4	1	2	5	3	2	3	6	3

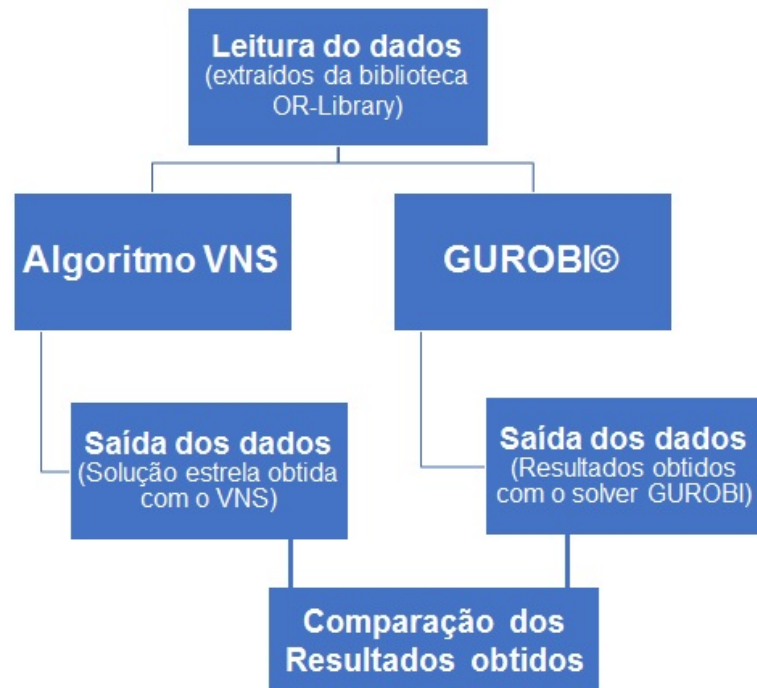
BIN 1	4	1								5
BIN 2	2	3								5
BIN 3	5									5
BIN 4	3	2								5
BIN 5	6									6
BIN 6	3									3

Fonte: Autoria Própria

3. METODOLOGIA

A metodologia apresenta as etapas seguidas durante o desenvolvimento do trabalho. Neste estudo, estará dividido entre a leitura dos dados, o algoritmo, e a exemplificação.

Figura 5: Etapas do desenvolvimento do trabalho.



Fonte: Autoria Própria

3.1. LEITURA DOS DADOS E SAÍDA DOS DADOS

O primeiro problema encontrado durante a leitura dos dados foi a maneira como as instâncias eram divididas no arquivo de entrada, pois cada instância possui 20 problemas. Para a leitura dos dados a serem utilizadas na meta-heurística, não houve problemas, pois ao programar o código foi possível fazê-lo entender que cada instância eram 20 problemas que precisariam ter 20 saídas. Porém, a utilização da mesma lógica de programação para a leitura dos dados no software de otimização GUROBI® não foi possível, pois cada vez que ele solucionava um problema ele reescrevia a solução sob a solução do problema anterior.

Portanto, a leitura dos dados foi realizada de maneiras diferentes, para cada tipo de análise. Na meta-heurística utilizou-se como obtido na biblioteca OR-Library

(<http://people.brunel.ac.uk/~mastjjb/jeb/info.html>) e no GUROBI© dividiu-se cada instância e formou-se 20 problemas diferentes. Logo, se eram 8 instâncias com 20 problemas cada uma, obteve-se 160 problemas diferentes.

3.2. O ALGORITMO

Após a leitura dos dados e o armazenamento destes em estruturas que já facilitariam o desenvolvimento do trabalho, aprofundou-se os estudos no algoritmo a ser implementado, o da Busca Vizinhança Variável. Com o embasamento teórico, já estava claro que o procedimento da meta-heurística em uso, iniciando não através de uma trajetória, mas sim explorando as vizinhanças gradativamente mais “distantes” da solução corrente assim focalizando a busca em torno de uma nova solução se e somente se um movimento de melhora foi realizado, ou seja, melhor solução até o momento também chamada de solução estrela.

Se a análise mostrar que o vizinho obteve melhores resultados, este vizinho passa a ser a solução estrela atual, em seguida se inicia uma nova iteração, ou seja, um novo passo é dado.

De acordo com SOUZA(2008), o pseudocódigo de um algoritmo de Busca Vizinhança Variável básico é similar a este apresentado abaixo:

Figura 6: Pseudocódigo Meta-heurística Busca em Vizinhança Variável

procedimento VNS()

```

1 Seja  $s_0$  uma solução inicial;
2 Seja  $r$  o número de estruturas diferentes de vizinhança;
3  $s \leftarrow s_0$ ; {Solução corrente}
4   enquanto (Critério de parada não for satisfeito) faça
5      $k \leftarrow 1$ ; {Tipo de estrutura de vizinhança corrente}
6     enquanto ( $k \leq r$ ) faça
7       Gere um vizinho qualquer  $s' \in \tau^{(?)}(s)$ ;
8        $s'' \leftarrow \text{BuscaLocal}(s')$ ;
9       se ( $f(s'') < f(s)$ )
10         então
11            $s \leftarrow s''$ ;
12            $k \leftarrow 1$ ;
13         senão
14            $k \leftarrow k + 1$ ;
15       fim-se;
16     fim-enquanto;
17   fim-enquanto;
18 Retorne  $s$ ;
19 fim VNS;
```

Fonte: Souza (2008)

De acordo com SILVA *et al.* (2000) os passos de um VNS básico são:

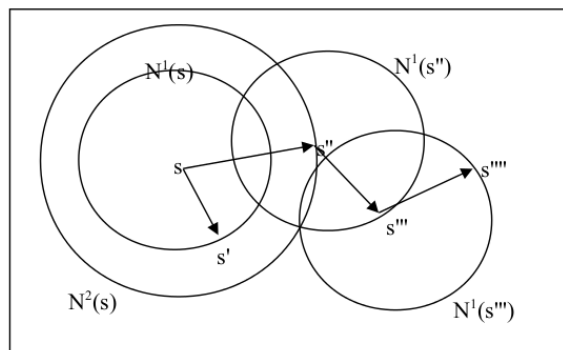
Seleção de um conjunto $kmáx$ de estruturas de vizinhanças N_k , ($k=1,2,\dots,kmáx$); Escolha de uma solução inicial x é um critério de parada; Partindo de uma solução inicial, analisa-se a primeira vizinhança $N_1(x)$; Caso numa vizinhança N_k não tenha atualizado a melhor solução, passa-se a analisar a próxima vizinhança N_{k+1} , caso contrário, se em N_k a melhor solução obtida até o momento for atualizada, retorna-se a vizinhança N_1 . O critério de parada é acionado quando no último $N_{kmáx}$

não se obteve nenhuma melhoria.

Para facilitar o entendimento de uma meta-heurística VNS, REIS (2013) propõe o seguinte modelo representado na Figura 7. Nesta figura pode-se observar como a meta-heurística VNS explora o espaço de soluções, em um problema de minimização. Partindo de uma solução corrente s presente dentro uma vizinhança $N^1(s)$ é realizado um movimento de exploração da vizinhança com intuito de encontrar uma solução s' que permita o escape das melhores soluções locais. A partir desta solução vizinha s' é realizada uma busca local na vizinhança da solução s' . Caso a melhor solução vizinha de s' não resulte em melhoria, retorna-se à solução s . Ao retornar a solução s o processo se reinicia, e uma solução s'' é escolhida dentro de uma estrutura de vizinhança $N^2(s)$, se esta solução apresentar melhoria ela seria aceita e a partir dela uma solução s''' será escolhida.

Sempre que uma solução de melhora for encontrada pela meta-heurística VNS, a solução corrente é movida para a nova solução e a busca é reiniciada, considerando a primeira estrutura de vizinhança (N^1). E na meta-heurística VNS, a busca local sempre é realizada na vizinhança de uma solução vizinha (escolhida aleatoriamente) da solução corrente.(REIS, 2013).

Figura 7: Funcionamento do VNS



Fonte: Reis (2013)

13.3 EXEMPLO

Na Tabela 1, temos o problema 1 da instância 1, onde os 120 *bins* estão sendo apenas

listados de forma aleatória, para que posteriormente sejam organizados de forma a minimizar o número de *bins*, no caso deste exemplo caminhões.

Tabela 1: Instância Bin Packing Problem 1

42	69	67	57	93	50	55	27	30	36
70	59	23	57	72	71	80	76	83	41
41	49	55	98	80	78	73	62	36	44
90	38	36	45	42	20	78	47	26	45
46	30	64	29	74	43	84	60	55	49
32	25	38	82	30	94	69	32	96	70
85	33	82	73	49	43	73	85	81	79
35	39	57	84	62	84	58	78	25	80
74	69	33	42	28	78	70	23	42	87
84	86	92	46	38	73	84	37	93	91
33	79	27	57	44	41	58	98	91	96
60	42	43	43	39	58	66	83	24	98

Fonte: Autoria Própria

Na Tabela 2 já se encontra a FO esperada obtida através da resolução do modelo matemático proposto no referencial teórico. O resultado esperado são 48 caminhões.

Tabela 2: Restrições e Resultados Esperados

Número de Bins	120
Capacidade da Frota	150
FO	48

Fonte: Autoria Própria

As Tabelas 3 e 4 exemplificam um dos problemas de agrupamento de tarefas em uma frota de veículos de umas das instâncias propostas para estudo neste trabalho. Podemos pensar nos *bins* heterogêneos como cargas de pesos diferentes, ou áreas diferentes que devem ser alocados em um caminhão sem que se ultrapasse a capacidade do mesmo. Contudo, é importante ressaltar que o número de caminhões deve ser minimizado a fim de reduzir os custos com este deslocamento.

Tabela 3: Exemplo de formação de frotas

Veículo 1	43+49+58=150
Veículo 2	43+49+58=150
Veículo 3	74+76=150
Veículo 4	72+78=150

Fonte: Autoria Própria

Na Tabela 4 estão representados apenas 4 carregamentos de caminhões como modelo de exemplificação. Cada conjunto de bloquinhos azuis deve ser visto como um veículo que será carregado com $n^o = x$ de itens.

Tabela 4: Representação de formação de frotas

20	23	23	24	25	25	26	27	27	28
29	30	30	30	32	32	33	33	33	35
36	36	36	37	38	38	38	39	39	41
41	41	42	42	42	42	42	43	43	43
43	44	44	45	45	46	46	47	49	49
49	50	55	55	55	57	57	57	57	58
58	58	59	60	60	62	62	64	66	67
69	69	69	70	70	70	71	72	73	73
73	73	74	74	76	78	78	78	78	79
79	80	80	80	81	82	82	83	83	84
84	84	84	84	85	85	86	87	90	91
91	92	93	93	94	96	96	98	98	98

Fonte: Autoria Própria

A Tabela 2 mostrou quais são as restrições e a função objetiva esperada para este problema. Sendo assim respeitando a capacidade da frota de 150, a solução ideal foi a formação de 48 frotas. A Tabela 5 representa este carregamento e as respectivas lotações de cada frota formada. Deve-se interpretar a tabela como cada cor representando uma formação de frota diferente, seguindo o raciocínio exemplificado na Tabela 3.

Tabela 5: Carregamento completo da frota para o problema 1

Número do veículo e seu respectivo carregamento										
1	148	9	149	17	150	25	150	33	149	41
2	149	10	150	18	150	26	149	34	150	42
3	150	11	148	19	149	27	149	35	149	43
4	150	12	150	20	150	28	150	36	150	44
5	149	13	150	21	150	29	150	37	148	45
6	150	14	150	22	150	30	150	38	149	46
7	150	15	150	23	150	31	150	39	150	47
8	150	16	150	24	150	32	150	40	144	48

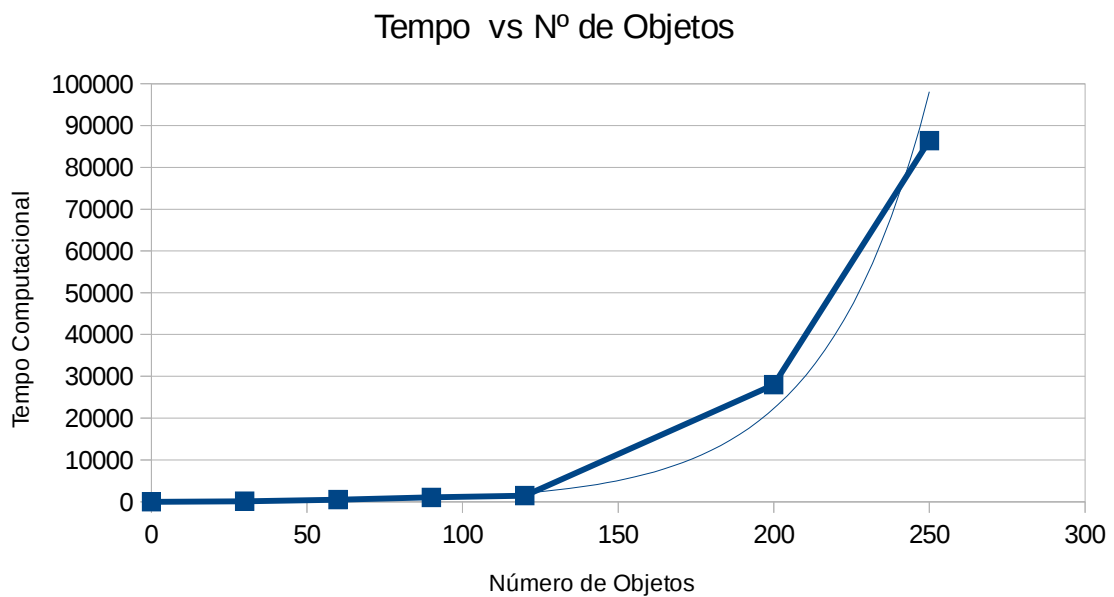
20	23	23	24	25	25	26	27	27	28
29	30	30	30	32	32	33	33	33	35
36	36	36	37	38	38	38	39	39	41
41	41	42	42	42	42	42	43	43	43
43	44	44	45	45	46	46	47	49	49
49	50	55	55	55	57	57	57	57	58
58	58	59	60	60	62	62	64	66	67
69	69	69	70	70	70	71	72	73	73
73	73	74	74	76	78	78	78	78	79
79	80	80	80	81	82	82	83	83	84
84	84	84	84	85	85	86	87	90	91
91	92	93	93	94	96	96	98	98	98

Fonte: Autoria Própria

24. ANÁLISE DOS RESULTADOS OBTIDOS

Após a realização de testes computacionais pôde-se comprovar através do um gráfico, representado pela Figura 8, que o problema VSBPP é NP-Hard. Isso se deve a natureza exponencial do modelo que soluciona o problema. Isto fica evidenciado em processamentos em que com mesmo número de objetos os tempos de processamento variam de 10 segundos a mais de 3 horas. Para a representação gráfica foi utilizado problemas com números de objetos variando entre 30, 60, 90, 120, 200, 250.

Figura 8: Gráfico Tempo vs Número de Objetos



Fonte: Autoria Própria

Na Tabela 6 estão representados os resultados obtidos na resolução dos 20 problemas da instância 1, tanto os resultados obtidos com a utilização da meta-heurística VNS quanto os resultados obtidos como solver GUROBI®.

Tabela 6: Resultados obtidos para os 20 problemas da instância 1

Problema	Solução Desejada	Solução obtida pelo VNS	Tempo VNS (s)	Solução obtida Pelo GUROBI	Tempo GUROBI (s)
0	48	50	22,95	48	81
1	49	51	23,11	49	1.575
2	46	48	10,45	46	24
3	49	51	42,84	ACIMA DE	50.575
4	50	52	79,87	50	1.701
5	48	50	247,04	48	20
6	48	50	123,02	48	4.795
7	49	52	27,78	ACIMA DE	10.949
8	51	53	75,66	ACIMA DE	11.107
9	46	48	490,01	ACIMA DE	22.941
10	52	54	45,64	52	3.369
11	49	51	37,88	ACIMA DE	11.954
12	48	51	26,29	ACIMA DE	11.654
13	49	51	51,05	49	10
14	50	52	20,08	50	613
15	48	50	35,89	48	12.800
16	52	55	11,94	52	5.749
17	52	55	37,77	ACIMA DE	14.073
18	49	51	26,98	ACIMA DE	22.246
19	50	52	39,16	ACIMA DE	12.185
		Tempo Total	1475,41	Tempo total	198.421

Fonte: Autoria Própria

Supondo que para os problemas em que o tempo de processamento ultrapassou as 3 horas, o solver GUROBI© tivesse conseguido obter todas as soluções no tempo computacional analisado, ainda sim, a somatória do tempo em horas seria de aproximadamente 55 horas, ao passo que o total de horas obtido com a meta-heurística foi de aproximadamente 25 minutos. Logo, é possível afirmar que mesmo que a meta-heurística VNS não retorne uma solução ótima, esta retorna uma solução viável em um tempo computacional relativamente baixo.

Utilizando o exemplo presente em 3.3, a meta-heurística VNS retornou uma solução viável de 50 veículos, o que é bem próximo ao valor esperado de 48. Ao analisar apenas um problema, é sempre melhor tentar encontrar o método que retorna a solução estrela, porém em uma empresa onde a frota de veículos e o número de problemas ultrapassam os 160 problemas estudados, onde a solução através de uma meta-heurística já se mostrou necessária, é possível

afirmar a necessidade de se estudar outros métodos de solução mais eficientes e eficazes.

A Tabela 6 resume todas as soluções encontradas com a utilização do software de otimização e com a utilização da meta-heurística VNS. Porém ao analisar o gráfico da Figura 8, uma pergunta surgiu porque uma amostra semelhante continuava apresentando valores discrepantes. Talvez, respondendo essa pergunta seria possível reduzir o tempo de processamento de alguns problemas.

Voltando a analisar o banco de dados, iniciou-se uma análise estatística dos resultados e a realização de novos testes com intuito de desvendar essa incógnita. Para realizar esta análise novos softwares foram usados como auxiliares, ferramenta *Input Analyser* do Arena, Software Estatístico Rstudio©. O *Input Analyser* do Arena, auxiliou na criação de gráficos com ajuste de melhor distribuição estatística das amostras, estes resultados estão resumidos na Tabela 7. Os tipos de distribuição resultantes foram tipo BETA e UNIFORME. Porém, esta informação não permitiu identificar alguma correlação, uma vez que tanto os problemas em amarelo, aqueles que excederam o tempo computacional viável, quanto os problemas em branco, que foram processados sem problemas, tiveram distribuições BETA e UNIFORME. Com isso, conclui-se que a distribuição estatística da amostra de dados não interferia na discrepância dos resultados obtidos.

Tabela 7: Resultados obtidos utilizando o Input Analyser, os problemas em amarelo são os que ultrapassarem as três horas no tempo de processamento

	P1	P2	P3	P4
Distribuição	BETA	Uniforme	BETA	Uniforme
Expressão	$19.5 + 79 * \text{BETA}(0, 0)$	$\text{UNIF}(0, 0)$	$19.5 + 81 * \text{BETA}(0.994, 1.14)$	$\text{UNIF}(0, 0)$
Erro Quadrático	0.009208	0.009043	0.007574	0.005849
Numero de intervalos	11	11	11	11
Graus de liberdade	8	10	8	10
Teste Estatístico	8.14	9.22	9.64	5.72
Correspondente valor p	0,43	0,511	0,301	0,75
Média	59,0	60,0	56,6	60,7
Desvio Padrão	22,2	23,3	23,1	24,1
	P5	P6	P7	P8
Distribuição	BETA	BETA	Uniforme	BETA
Expressão	$19.5 + 80 * \text{BETA}(0.934, 0.9)$	$19.5 + 81 * \text{BETA}(0.929, 0.959)$	$\text{UNIF}(19.5, 101)$	$19.5 + 81 * \text{BETA}(1.32, 1.27)$
Erro Quadrático	0.009452	0.008041	0.008904	0.006800
Numero de intervalos	11	11	11	11
Graus de liberdade	8	8	10	8
Teste Estatístico	11.8	5.52	13.8	11.7
Correspondente valor p	0,174	0,701	0,193	0,178
Média	61,3	59,4	59,5	60,8
Desvio Padrão	23,6	23,8	23,3	21,4
	P11	P12	P11	P12
Distribuição	BETA	BETA	BETA	BETA
Expressão	$19.5 + 81 * \text{BETA}(1.16, 0.944)$	$20.5 + 80 * \text{BETA}(0.904, 0.909)$	$19.5 + 81 * \text{BETA}(1.16, 0.944)$	$20.5 + 80 * \text{BETA}(0.904, 0.909)$
Erro Quadrático	0.007804	0.009655	0.007804	0.009655
Numero de intervalos	11	11	11	11
Graus de liberdade	8	8	8	8
Teste Estatístico	3.98	12.8	3.98	12.8
Correspondente valor p	0,75	0,127	0,75	0,127
Média	64,0	60,4	64,0	60,4
Desvio Padrão	23,1	23,8	23,1	23,8
	P13	P14	P15	P16
Distribuição	BETA	BETA	Uniforme	BETA
Expressão	$19.5 + 80 * \text{BETA}(1.35, 1.33)$	$19.5 + 81 * \text{BETA}(0.917, 0.917)$	$\text{UNIF}(20.5, 101)$	$19.5 + 81 * \text{BETA}(0.876, 0.888)$
Erro Quadrático	0.006634	0.006997	0.007361	0.006204
Numero de intervalos	11	11	11	11
Graus de liberdade	8	8	10	8
Teste Estatístico	6.45	1.7	9.66	3.45
Correspondente valor p	0,598	0,75	0,475	0,75
Média	59,8	60,0	61,4	59,2
Desvio Padrão	20,8	24,1	22,9	24,8
	P17	P18	P19	P20
Distribuição	BETA	BETA	Uniforme	BETA
Expressão	$20.5 + 80 * \text{BETA}(1.07, 0.897)$	$21.5 + 79 * \text{BETA}(1.27, 1.08)$	$\text{UNIF}(20.5, 101)$	$20.5 + 80 * \text{BETA}(0.908, 0.867)$
Erro Quadrático	0.006690	0.007289	0.006944	0.006994
Numero de intervalos	11	11	11	11
Graus de liberdade	8	8	10	8
Teste Estatístico	2.53	7.32	7.31	7.48
Correspondente valor p	0,75	0,502	0,695	0,488
Média	64,1	64,2	60,5	61,0
Desvio Padrão	23,1	21,5	22,8	24,4

Fonte: Autoria Própria

Com o Rstudio© analisou-se coeficiente de correlação, que indica a força e a direção do relacionamento linear entre duas variáveis aleatórias. O valor do coeficiente deveria estar próximo de -1 para correlação negativa ou próximo de 1 para correlação positiva. Todos os valores encontrados demonstrados Tabela 8 se aproximavam de zero, isto significa que as duas variáveis não dependem linearmente uma da outra. No entanto, pode existir uma dependência não linear que deve ser investigada por outros meios.

Tabela 8: Matriz de correlações de Pearson para Instância 1, matriz obtida através do software Rstudio©.

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20
P1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0,048	1	0,149	-0,19	-0,02	0,014	0,074	-0	0,027	0,043	-0,14	-0,23	0,095	-0,09	0,072	-0,13	-0,19	0,02	0,057	0,085
P3	0,004	0,149	1	-0,06	0,038	-0,11	-0,02	0,182	0,051	0,124	-0,02	-0,14	-0,12	-0,01	-0,01	0,005	-0,14	0,108	0,082	0,111
P4	-0	-0,19	-0,06	1	-0,03	0,009	0,008	0,011	-0,04	-0,03	-0,05	0,047	0,131	0,119	0,091	0,065	-0,07	-0,03	0,141	-0,02
P5	0,013	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P6	0,033	0,014	-0,11	0,009	0,126	1	0,082	-0,21	0,105	-0	0,086	0,094	-0,02	-0,11	0,058	-0,02	-0,06	0,009	0,118	0,083
P7	-0,03	0,074	-0,02	0,008	-0,02	0,082	1	0,117	0,148	-0,1	-0,15	-0,11	0,081	0,282	0,104	-0,07	-0,1	-0,06	0,156	-0,03
P8	-0,17	-0	0,182	0,011	0,059	-0,21	0,117	1	-0,04	-0,05	-0,14	-0,01	0,134	0,118	-0,03	-0,1	-0,04	0,132	-0	-0,04
P9	0,014	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
P10	0,006	0,043	0,124	-0,03	0,094	-0	-0,1	-0,05	0,137	1	0,099	0,14	0,04	-0,18	0,003	0,11	0,037	-0,07	-0,05	0,073
P11	0,158	-0,14	-0,02	-0,05	0,073	0,086	-0,15	-0,14	-0,05	0,099	1	-0,04	-0,05	0,085	0,072	-0	-0,05	-0,03	-0,02	0,036
P12	-0,11	-0,23	-0,14	0,047	0,154	0,094	-0,11	-0,01	0,098	0,14	-0,04	1	0,021	-0,06	0,046	0,28	0,141	0,017	-0,11	-0,05
P13	-0,1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
P14	0,069	-0,09	-0,01	0,119	-0,06	-0,11	0,282	0,118	-0,02	-0,18	0,085	-0,06	-0,09	1	0,084	-0,06	-0	-0,08	0,069	-0,11
P15	0,068	0,072	-0,01	0,091	0,096	0,058	0,104	-0,03	-0,14	0,003	0,072	0,046	0,011	0,084	1	0,077	-0,02	-0,14	0,061	0,187
P16	-0,01	-0,13	0,005	0,065	0,187	-0,02	-0,07	-0,1	0,067	0,11	-0	0,28	0,084	-0,06	0,077	1	0,091	0,008	-0,05	0,103
P17	0,044	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
P18	-0,15	0,02	0,108	-0,03	-0,09	0,009	-0,06	0,132	-0,09	-0,07	-0,03	0,017	0,103	-0,08	-0,14	0,008	0,028	1	-0,09	-0,05
P19	-0,14	0,057	0,082	0,141	-0,03	0,118	0,156	-0	-0,03	-0,05	-0,02	-0,11	0,042	0,069	0,061	-0,05	-0,04	-0,09	1	0,041
P20	0,077	0,085	0,111	-0,02	0,096	0,083	-0,03	-0,04	0,003	0,073	0,036	-0,05	0,111	-0,11	0,187	0,103	0,092	-0,05	0,041	1

Fonte: Autoria Própria

Após descartar a possibilidade de correlação linear, uma nova ideia surgiu baseado na ideia do algoritmo FFD, “ Com a ordenação dos dados, é possível existir alguma melhora nos tempos computacionais?”. Esta pergunta desencadeou novos testes em instâncias menores para averiguar essa nova possibilidade. Desta vez os testes foram realizados em dois pacotes de otimização no LINGO© e no GUROBI©.

A Tabela 9 demonstra os resultados computacionais obtidos nos testes com a utilização do LINGO©. Com ela, é possível concluir que a ordenação decrescente dos dados resulta em melhores tempos computacionais.

Tabela 9: Tempos computacionais obtidos nos testes utilizando LINGO©.

Lingo Resultados (segundos)	Ordenação Normal	Ordenação Crescente	Ordenação Decrescente	Solução Ótima
Problema 1	11	90	5	22
Problema 2	3	3	3	25
Problema 3	5	638	5	24
Problema 4	755	19	7	20
Problema 5	+ que 10800	2408	1716	21
Problema 6	356	28	11	24

Fonte: Autoria Própria

Na Tabela 10, o que se encontra é outra análise, a ordenação crescente dos dados no GUROBI© resulta em melhores tempos computacionais. Para saber o porque ocorre essas diferenças, ou seja, porque a ordenação decrescente é mais indicada para o LINGO© e a ordenação crescente é mais indicada para o GUROBI© é preciso entender a maneira com quem esses softwares foram programados.

Tabela 10: Tempos computacionais obtidos nos testes utilizando GUROBI©.

GUROBI Resultados	Ordenação Normal	Ordenação Crescente	Ordenação Decrescente	Solução Ótima
Problema 1	479	7	30	22
Problema 2	525	18	43	25
Problema 3	398	5	27	24
Problema 4	644	28	62	20
Problema 5	567	23	45	21
Problema 6	436	6	67	24

Fonte: Autoria Própria

De maneira geral, a conclusão que se obteve foi: ao analisar ambas tabelas é possível concluir que independente se ordenado crescente ou decrescente, ao ordenar os dados os dois softwares apresentaram uma melhora nos tempos computacionais. Portanto, a ordenação de dados é uma solução viável quando há necessidade de melhoria nos tempos computacionais para problemas *NP-hard*.

5. CONSIDERAÇÕES FINAIS

Acredita-se que desde o homem pré-histórico os seres humanos tentam solucionar os problemas que surgiam no dia a dia por meio do raciocínio lógico, muito antes da escrita o homem já sabia identificar as quantidades dos objetos, contar o tempo, assim se desenvolvia a matemática. Atualmente grandes partes dos problemas cotidianos são solucionados através de formulações matemáticas. Para auxiliar a tomada de decisão podemos utilizar a teoria da otimização, como sendo o trabalho de identificar em um universo de soluções possíveis, qual é a melhor, ou seja, a solução ótima para um problema específico.

Sendo assim, este projeto deu continuidade a uma linha de pesquisa que se baseia na aplicação das técnicas de pesquisa operacional para a resolução de problemas típicos da engenharia de produção. Foi implementado uma abordagem meta-heurística para o problema de agrupamento de entregas em veículos heterogêneos. A implementação foi validada com a resolução de problemas de instâncias de testes *benchmarking* da literatura e com os resultados obtidos através da utilização do software de otimização GUROBI®.

A leitura dos dados, mesmo que adaptada para os diferentes problemas, foi realizada com sucesso, a implementação da meta-heurística VNS também. O problema foi o tempo computacional excedido ao utilizar pacotes de otimização na maioria dos problemas. A partir deste problema, iniciou-se uma análise estatística dos resultados e a realização de novos testes com intuito de desvendar a incógnita, do motivo de um banco de dados com dados semelhantes apresentar resultados discrepantes. Após a utilização de diferentes softwares de análise estatística não foi possível afirmar qual a correlação da amostra dos dados, pode-se afirmar que as duas variáveis não dependem linearmente uma da outra e que a distribuição estatística da amostra não interfere nos resultados. No entanto, pode existir uma dependência não linear que deve ser investigada em futuros trabalhos. Com a realização de novos testes, foi possível afirmar que a ordenação da amostra de dados deverá reduzir o tempo computacional excedido.

A programação computacional, acompanhada de um modelo matemático com objetivos bem definidos deve ser sempre indicada como ferramenta de auxílio na tomada de decisão. Se implementado corretamente o programa oferecerá alta confiabilidade e praticidade. Dito isto, podemos afirmar que os objetivos foram alcançados. Foi desenvolvido

um modelo matemático que é capaz de atender às necessidades de casos reais. Acumulou-se conhecimento a cerca dos métodos de resolução do *Bin Packing Problem*, e por fim espera-se com este trabalho difundir a utilização de técnicas de otimização na operação do sistema de transporte, possibilitando a redução de seus custos, uma vez que a redução do números de veículos a serem utilizados automaticamente reduz os custos com transporte de uma empresa. O próximo passo será testar e validar o modelo com dados reais de uma empresa que opera no território nacional.

6. REFERÊNCIAS

ALVIN, A. C. F., **Uma Heurística Híbrida de Melhoria para o Problema de Bin Packing e sua Aplicação ao Problema de Escalonamento de Tarefas**. Tese de doutorado, Programa de Pós –graduação em Informática, PUC – Rio, Rio de Janeiro, junho de 2003.

ANDERSON, RICHARD J.; MAYR, ERNST W.; WARMUTH, MANFRED K. Parallel approximation algorithms for Bin Packing. **Information and Computation**, v. 82, n. 3, p. 262-277, 1989.

ARENALES, M. N.; ARMENTANO, V.; MORABITO, R. **Pesquisa Operacional**. Rio de Janeiro: Campus, 2006.

ARENALES, M.; ARMENTANO, V. MORABITO. R. YANASSE, H. **Pesquisa operacional**, v. 6, 2007.

CORREIA, ISABEL; GOUVEIA, LUÍS; SALDANHA-DA-GAMA, FRANCISCO. Solving the variable size Bin Packing problem with discretized formulations. **Computers & Operations Research**, v. 35, n. 6, p. 2103-2113, 2008.

DIAZ, A. OPTIMIZACIÓN HEURÍSTICA Y REDES NEURONABLES. **Editorial Paraninfo**, Madrid, 1996.

DORIGO, MARCO; STÜTZLE, THOMAS. The ant colony optimization metaheuristic: Algorithms, applications, and advances. **In: Handbook of metaheuristics**. Springer US, 2003. p. 250-285.

FLEURY, PAULO FERNANDO. Gestão estratégica do transporte. **Revista Tecnológica**, n. 82, 2002.

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

HILLIER, F. S.; LIEBERMAN, G. J. **Introdução a Pesquisa Operacional**, 8. Ed. Mcgraw-Hill, 2010.

LIMA, Maurício. Custos logísticos no Brasil, 2014. Disponível em:< <http://www.ilos.com.br/web/custos-logisticos-no-brasil />>. Acesso em: 6 de mar. 2016.

MELIÁN, B. et al. Metaheuristics: A Global View. In Revista Iberoamericana de Inteligencia Artificial. **Asociación Española de Inteligencia Artificial**, 2003. v. 2, n. 19

MIURA,M. **Modelagem Heurística no Problema de Distribuição de Cargas Fracionadas de Cimento**. Dissertação (Mestrado) – Escola Politécnica da Universidade de

São Paulo. Departamento de Engenharia de Transportes. Sistemas Logísticos. São Paulo, 2008.

MOORE, DAVID S. (2007), **The Basic Practice of Statistics**. New York, Freeman.

PINEDO, MICHAEL; CHAO, XIULI. **Operations scheduling**. McGraw Hill, 1999.

REIS, J. A.; CUNHA, C. B. (2007), **Uma heurística baseada em busca em vizinhança variável para o problema de agrupamento de entregas em veículos de uma frota heterogênea**. Pós-Graduação em Engenharia de Transportes, Escola Politécnica da USP, São Paulo, 2007.

REIS, JORGE VON ATZINGEN. **Meta-heurísticas baseadas em busca em vizinhança variável aplicadas a problemas de operação de transportes**. 2013. Tese de Doutorado. Universidade de São Paulo.

ROSA, A. C. (2007), **Gestão do transporte na logística de distribuição física: Uma análise da minimização do custo operacional**. Dissertação para mestrado pelo Curso de Gestão em Desenvolvimento Regional do Departamento de Economia, Contabilidade e Administração da Universidade de Taubaté, São Paulo, 2007.

RIBEIRO, P. C. C.; FERREIRA, K. A. (2002), Logística e transportes: uma discussão sobre os modais de transporte e o panorama brasileiro. **XXII Encontro Nacional de Engenharia de Produção – ENEGEP** (2002).

SANTOS, Antônio Marcos dos; RODRIGUES, Iana Araújo. **Controle de estoque de materiais com diferentes padrões de demanda: estudo de caso em uma indústria química**. Gest. Prod., São Carlos, v. 13, n. 2, p. 223-231, May 2006.

SANTOS, C. H. S.; BASSANESSI, M. M. R. (2006) **O futuro da atividade Logística. Logística e Distribuição Física Internacional: Teoria e Pesquisas**. Aduaneiras, São Paulo, p. 11-24.

SILVA, M. B.; DRUMMOND, LMA; OCHI, L. S. Metaheurísticas grasp+ vns para a solução de problemas de otimização combinatória. **In: Congresso brasileiro de pesquisa operacional**. 2000.

SILVA, J.L.C.; SOMA, N.Y. Um algoritmo polinomial para o problema de empacotamento de contêineres com estabilidade estática da carga pesquisa. **Operacional**, v. 23, n. 1, p.79-98, Jan./Abr. 2003.

SOUZA, MARCONE JAMILSON FREITAS. **Inteligência computacional para otimização**. Notas de aula, Departamento de Computação, Universidade Federal de Ouro

Preto 2008.

VEL LEJBMAN, DR ALFREDO GOLDMAN. **Um Estudo Abrangente sobre Metaheurística, incluindo um Histórico.** 2009.

WALTER, C.: **Planejamento e Controle da Produção - PCP.**[S.l.:s.n.], Apostila de aula.-Mestrado em Engenharia de Produção - UFRGS, 1999.

ANEXO A – CÓDIGO UMA META-HEURÍSTICA VNS APLICADA AO PROBLEMA VSBPP.

```
//-----
//Inclusão das bibliotecas
//-----
#include <iostream> //cin, cout
#include <fstream> //ifstream, ofstream
#include <vector> //vector
#include <time.h> //clock_t
#include <iomanip> //setprecision
#include <stdlib.h> //srand
#include <string> //Facilitar a escolha da leitura de arquivos
using namespace std;
//-----
//Declaração das variáveis globais
//-----
#define max 120
//-----
//Declaração das variáveis globais
//-----
class bin
{
public:
    int codigo;
    float capacidade;
    int nitens;
    int solucao;
};

//Variáveis usadas na leitura de dados
int nproblemas;
```

```
vector <vector<float> > peso;    //Vetor que armazena o peso dos
objetos lidos de cada problema
```

```
vector <float> consumo_cap;      //Vetor que auxilia o
preenchimento do vetor peso
```

```
vector <bin> caminhao;          //Vetor que armazenará os dados
lidos
```

```
string num_arq;                //Armazena o numero do arquivo que será
resolvido
```

```
//Variáveis usadas na ordenação
```

```
float A[max];                  //auxilia a ordenação do vetor peso
```

```
time_t tempo;                  //Contador de tempo computacional
```

```
vector<time_t> tempo_resposta; //Medir o tempo necessário para
a resolução de um problema
```

```
//Variáveis usadas na FO e na solução inicial
```

```
vector<int> numbins;           //Armazena quantos bins
serão necessários para o trsnporte
```

```
vector<vector<vector<int> > > resposta;    //Armazena a
solução final do problema
```

```
vector<int> respfinal;         //Armazena o número final de
caminhões necessário para resolver o problema
```

```
//Variáveis utilizadas nas estruturas de vizinhança
```

```
float contador=0;              //Controla a troca de estruturas de
vizinhança
```

```
//Variáveis usadas na impressão e na função principal
```

```
int escolha;
```

```
//-----
//Declaração das funções
//-----

void leitura();
void impressao();
void sol_ini(int auxilio);
void sol_ini2(int auxilio);
void quicksort(float A[max], int esq, int dir);
void criacao_resposta();
void preenche_matriz(int auxilio);
void vns1(int auxilio);
void vns2(int auxilio);
void FO(int auxilio);
//-----
//int main(): Função Principal
//-----
int main()
{
    int validacao;

    leitura();
    //Ordena os itens por peso
    tempo_resposta.resize(0);
    for(int k=0;k<nproblemas;k++)
    {
        tempo= clock();
        for(int n=0;n<max;n++)
            A[n]= peso[k][n];
        quicksort(A,0,max-1);
        for(int n=0;n<max;n++)
```

```

        peso[k][n]= A[n];
        tempo=clock() -tempo;
        tempo_resposta.push_back(tempo);
    }

//Escolha da ação a ser realizada
do
{
    //Escolha do tipo de solução inicial
    cout<<"\n*****\n";
    cout<<"1 - Para solucao inicial gulosa\n";
    cout<<"2 - Para teste nas estruturas de vizinhanca\n";
    cout<<"*****\n";
    cin>>escolha;
    cout<<"\n";
}while(escolha!=1 && escolha!=2);

//Calcula o número de bins necessários para a solução inicial
numbins.resize(nproblemas);
//Resolução por meio da inicial solução gulosa
if(escolha==1)
{
    for(int z=0;z<nproblemas;z++)
    {
        numbins[z]=0;
        sol_ini(z);
    }
    //Cria as matrizes de solução inicial e vizinha
    criacao_resposta();

    //Preenchendo a matriz solução inicial

```

```

    numbins.resize(nproblemas);
    for(int z=0;z<nproblemas;z++)
    {
        numbins[z]=0;
        tempo= clock();
        preenche_matriz(z);
        tempo= clock()-tempo;
        tempo_resposta.push_back(tempo);
    }
    cout<<"\n";

}

//Teste das estruturas de vizinhança
if(escolha==2)
{
    for(int z=0;z<nproblemas;z++)
    {
        if(z==0)
            resposta.resize(nproblemas);
        sol_ini2(z);
    }

}

//Estruturas de vizinhança
cout<<fixed;
srand(time(NULL));
for(int i=0;i<nproblemas;i++)
{
    cout<<"Resolvendo o problema numero "<<i+1<<"\n";

```

```

tempo= clock();
do
{
//      cout<<setprecision(2)<<contador<<"\n";
      if(contador<100)
          vns1(i);
      if(contador>=100)
          vns2(i);
}while(contador<2000);
contador=0;
tempo= clock() - tempo;
tempo_resposta.push_back(tempo);
}
//Atualiza a FO
respfinal.resize(nproblemas);
for(int i=0;i<nproblemas;i++)
{
    respfinal[i]= numbins[i];
    FO(i);
}

//Verifica se o problema foi resolvido corretamente
for(int i =0;i<nproblemas;i++)
{
    validacao=0;
    for(int j=0; j<numbins[i];j++)
    {
        for(int k=0;k<caminhao[i].nitens;k++)
            validacao+= resposta[i][j][k];
        if(validacao == caminhao[i].nitens)
            cout<<"O arquivo foi resolvido corretamente\n";
    }
}

```

```

    }
}
impressao();
}

//-----
//void leitura(): Le os dados do arquivo de entrada
//-----
void leitura()
{
    string nome_arq;
    //int ajuda;
    cout<<"Digite o numero do arquivo Binpacking que deseja
resolver: ";
    cin>>num_arq;

    nome_arq= "Binpack"+ num_arq + ".txt";
    ifstream origem (nome_arq.c_str());
    if (!origem)
        cerr<< "\nErro ao abrir o arquivo Binpack.txt\n\n";

    cout<<"Iniciando a leitura dos dados de entrada\n\n";
    bin aux;
    float aux2;

    caminhao.resize(0);
    peso.resize(0);

    origem>>nproblemas;

    for(int i=0;i<nproblemas;i++)

```

```

{
    //Leitura do objeto que armazena os dados para iniciar a leitura
dos objetos a serem alocados

    origem>> aux.codigo;
    origem>> aux.capacidade;
    origem>> aux.nitens;
    origem>> aux.solucao;
    caminhao.push_back(aux);    //Alocação dinâmica do vetor
caminhão

    //Preenchimento do vetor consumo_cap
    consumo_cap.resize(0);
    for(int j=0;j<aux.nitens;j++)
    {
        origem>> aux2;
        consumo_cap.push_back(aux2);
    }

    //Alocando o vetor consumo_cap no vetor peso
    peso.push_back(consumo_cap);
}

//-----
//Funcao quicksort: ordena os objetos por peso
//-----
void quicksort(float A[max], int esq, int dir)
{
    int i, j, pivo, aux;

    i= esq;

```

```

j= dir;
pivo= (esq+dir)/2;
pivo= A[pivo];

do
{
    while((A[i] < pivo) && (i < dir))
        i++;
    while((A[j] > pivo) && (j > esq))
        j--;
    if (i <= j)
    {
        aux= A[i];
        A[i]= A[j];
        A[j]= aux;
        i++;
        j--;
    }
} while( i <= j);
if (esq < j)
    quicksort(A, esq, j);
if (i < dir)
    quicksort(A, i, dir);
}

//-----
//Funcao sol_ini(): Calcula uma solução gulosa para o problema
//-----
void sol_ini(int auxilio)
{
    cout<<"Iniciando o calculo do numero de bins necessarios para o
problema "<<auxilio+1<<"\n";

```

```

int folga, cont=caminhao[auxilio].nitens-1, cont2=0;

do
{
    numbins[auxilio]++;
    folga= caminhao[auxilio].capacidade;
    while(folga-peso[auxilio][cont]>0 && cont>= cont2)
    {
        folga-=peso[auxilio][cont];
        cont--;
    }

    while(folga-peso[auxilio][cont2]>0 && cont>= cont2)
    {
        folga-=peso[auxilio][cont2];
        cont2++;
    }
}while(cont>=cont2);
}

//-----
//Funcao criacao_resposta(): Cria a matriz resposta para a soluão
inicial e final
//-----
void criacao_resposta()
{
    int cont=0;
    resposta.resize(nproblemas);
    do
    {
        resposta[cont].resize(numbins[cont]);

```

```

//Criação da matriz da resposta final
for(int i=0;i<numbins[cont];i++)
    resposta[cont][i].resize(caminhao[cont].nitens);
//Zerando a matriz
for(int j=0;j<numbins[cont];j++)
    for(int z=0;z<caminhao[cont].nitens;z++)
        resposta[cont][j][z]=0;
    cont++;
}while(cont<nproblemas);

cont=0;
}
//-----
//Funcao preencher_matriz(): preenche a matriz com a solução inicial
//-----
void preenche_matriz(int auxilio)
{
    int folga, cont=caminhao[auxilio].nitens-1, cont2=0;

    do
    {
        numbins[auxilio]++;
        folga= caminhao[auxilio].capacidade;
        while(folga-peso[auxilio][cont]>0 && cont>= cont2)
        {
            folga-=peso[auxilio][cont];
            resposta[auxilio][numbins[auxilio]-1][cont]++;
            cont--;
        }
        while(folga-peso[auxilio][cont2]>0 && cont>= cont2)
        {

```

```

        folga-=peso[auxilio][cont2];
        resposta[auxilio][numbins[auxilio]-1][cont2]++;
        cont2++;
    }
}while(cont>=cont2);
}
//-----
//Funcao impressao(): Imprime os principais dados junto com a
solução inicial
//-----
void impressao()
{
    string saida;
    saida= "Saida_Problema"+ num_arq + ".txt";
    ofstream destino (saida.c_str());
    if (!destino)
        cerr << "Arquivo Saida_Problema.txt nao pode ser aberto\n";
    for(int i=0;i<nproblemas;i++)
    {
        destino<<fixed;
        destino<<"\nO Problema resolvido foi: "<<setprecision(1)
<<caminhao[i].codigo<<"\n";
        destino<<"A capacidade do bin e de:
"<<caminhao[i].capacidade<<"\n";
        destino<<"O numero de itens a serem alocados e de:
"<<caminhao[i].nitens<<"\n";
        destino<<"A melhor solucao foi de:
"<<caminhao[i].solucao<<"\n";
        destino<<"A solucao inicial foi de: "<<numbins[i]<<"\n";
        destino<<"A solucao atual foi de: "<<respfinal[i]<<"\n";
        destino<<"O tempo para ordenacao dos itens foi de:

```

```

"<<setprecision(5)<<((float)tempo_resposta[i])/CLOCKS_PER_SEC<<"\n";
    if(escolha==1)
    {
        destino<<"O tempo gasto para gerar a solucao inicial foi de:
"<<setprecision(5)<<((float)tempo_resposta[i+nproblemas])/CLOCKS_PER_
SEC<<"\n";

        destino<<"O tempo gasto para gerar a solucao final foi de:
"<<setprecision(5)<<((float)tempo_resposta[i+nproblemas+nproblemas])/
CLOCKS_PER_SEC<<"\n";
    }
    if(escolha==2)
        destino<<"O tempo gasto para gerar a solucao final foi de:
"<<setprecision(5)<<((float)tempo_resposta[i+nproblemas])/CLOCKS_PER_
SEC<<"\n";
    /*    for(int j=0;j<numbins[i];j++)
    {
        destino<<"Os itens a serem alocados no caminhao
"<<j+1<<" sao: ";
        for(int k=0;k<caminhao[i].nitens;k++)
        {
            if(resposta[i][j][k]!=0)
                destino<<k+1<<" ";
            //destino<<peso[i][k]*resposta[i][j][k];
        }
        destino<<"\n";
    }
*/    }
    cout<<"\nO arquivo foi gerado com sucesso!!!\n";
    destino.close();
}
//-----

```

```

//Funcao vns1(): Primeira estrutura de vizinhança
//-----
void vns1(int auxilio)
{
    int num_iten[2], num_caminhao[2], folga[2];

    //Escolhe aleatoriamente um caminho e um iten que será
alocado em outro caminho
    for(int i=0;i<2;i++)
    {
        do
        {
            num_iten[i]= rand()%caminhao[auxilio].nitens;
            num_caminhao[i]= rand()%numbins[auxilio];
        }while(resposta[auxilio][num_caminhao[i]][num_iten[i]]== 0);
    }
    //Calcula a folga do caminho que receberá o item
    for(int i=0; i<2;i++)
        folga[i]=0;
    for(int i=0; i<2; i++)
        for(int j=0;j<caminhao[auxilio].nitens;j++)
            folga[i]+=resposta[auxilio][num_caminhao[i]]
[j]*peso[auxilio][j];

    for(int i=0;i<2;i++)
        folga[i]=caminhao[auxilio].capacidade -folga[i] + peso[auxilio]
[num_iten[i]];

    //Troca dos itens
    if(folga[0]-peso[auxilio][num_iten[1]]>=0  &&  folga[1]-
peso[auxilio][num_iten[0]]>=0)

```

```

{
    resposta[auxilio][num_caminhao[0]][num_iten[0]]--;
    resposta[auxilio][num_caminhao[1]][num_iten[0]]++;
    resposta[auxilio][num_caminhao[1]][num_iten[1]]--;
    resposta[auxilio][num_caminhao[0]][num_iten[1]]++;
}
contador++;
}
//-----
//Funcao vns2(): Segunda estrutura de vizinhança
//-----
void vns2(int auxilio)
{
    int num_iten, num_caminhao[2], folga, impedimento;

    //Escolhe aleatoriamente um caminhao e um iten que será
alocado em outro caminhao
    num_iten= rand()%caminhao[auxilio].nitens;

    for(int i=0;i<2;i++)
        num_caminhao[i]=-1;
    for(int i=0;i<2;i++)
    {
        do
            num_caminhao[i]= rand()%numbins[auxilio];
            while(resposta[auxilio][num_caminhao[0]][num_iten]== 0 ||
num_caminhao[0]==num_caminhao[1]);
    }

    //Calcula a folga do caminhao que receberá o item
    folga=0;

```

```

for(int j=0;j<caminhao[auxilio].nitens;j++)
    folga+=resposta[auxilio][num_caminhao[1]][j]*peso[auxilio][j];

folga=caminhao[auxilio].capacidade -folga;

impedimento=0;
for(int i=0; i<caminhao[auxilio].nitens;i++)
    impedimento+=resposta[auxilio][num_caminhao[1]][i];

if(folga-peso[auxilio][num_iten]>=0 && impedimento!=0)
{
    resposta[auxilio][num_caminhao[0]][num_iten]--;
    resposta[auxilio][num_caminhao[1]][num_iten]++;
    contador= 0;
}
else
    contador++;
}

//-----
//Funcao FO(): Atualiza o valor da FO
//-----

void FO(int auxilio)
{
    int cont=0;

    for(int k=0;k<numbins[auxilio];k++)
    {
        cont=0;
        for(int j=0;j<caminhao[auxilio].nitens;j++)
            cont+=resposta[auxilio][k][j];
        if(cont==0)

```


ANEXO B – IMPLEMENTAÇÃO DO GUROBI PARA O PROBLEMA VSBPP.

```

//-----
//Inclusão das bibliotecas
//-----
#include "gurobi_c++.h"
#include <iostream> //cin, cout
#include <fstream> //ifstream, ofstream
#include <vector> //vector
#include <time.h> //clock_t
#include <iomanip> // setprecision
using namespace std;
//-----
//Declaração das Constantes
//-----
#define N_ITEM 125
//-----
//Declaração das variáveis globais
//-----
int FO;
time_t t_ini, t_fim; //contadores de tempo computacional
double tempo; //contadores de tempo computacional
int respostay[N_ITEM], respostax[N_ITEM][N_ITEM];
//FILE *destino;
//char *nome = (char*) malloc(sizeof(char*));
class bin
{
public:
    int codigo;
    float capacidade;
    int nitens;
    int solucao;
};

```

```

vector <vector<float> > peso;
vector <float> consumo_cap;
vector <bin> caminhao;
//-----
//Declaração das funções
//-----

void leitura();
void impressao();
void gurobi();
//-----
//int main(): Função Principal
//-----
int main()
{
    leitura();
    gurobi();
    impressao();

}
//-----
//Leitura
//-----
void leitura()
{
    ifstream origem ("incbin11.txt");
    if (!origem)
        cerr<< "\nErro ao abrir o arquivo Bincpack11.txt\n\n";

    bin aux;
    float aux2;

```

```

caminhao.resize(0);
peso.resize(0);

origem>> aux.codigo;
origem>> aux.capacidade;
origem>> aux.nitens;
origem>> aux.solucao;
caminhao.push_back(aux); //Alocação dinâmica do vetor caminhão

//Preenchimento do vetor consumo_cap
consumo_cap.resize(0);
for(int j=0;j<aux.nitens;j++)
{
    origem>> aux2;
    consumo_cap.push_back(aux2);
}

//Alocando o vetor consumo_cap no vetor peso
peso.push_back(consumo_cap);
}
//-----
//Gurobi
//-----
void gurobi()
{
    time(&t_ini);
    GRBEnv* env = 0;
    GRBVar* y = 0;
    GRBVar** x = 0;

    try
    {

```

```

printf("\n-----\n\n");
printf("\nIniciando resolução do problema.\n\n");
env = new GRBEnv(); //inicia ambiente gurobi
GRBModel model = GRBModel(*env); //cria um modelo
model.set(GRB_StringAttr_ModelName, "Bin-packing");

//-----
//Dados de entrada
//-----
y = model.addVars(caminhao[0].nitens, GRB_BINARY);
model.update();
for (int j= 0; j< caminhao[0].nitens; j++)
{
    y[j].set(GRB_DoubleAttr_Obj, 1);
    y[j].set(GRB_StringAttr_VarName, "y");
}

x = new GRBVar* [caminhao[0].nitens];
for(int i= 0; i< caminhao[0].nitens; i++)
{
    x[i] = model.addVars(caminhao[0].nitens, GRB_BINARY);
    model.update();
    for (int j= 0; j< caminhao[0].nitens; j++)
    {
        x[i][j].set(GRB_DoubleAttr_Obj, 0);
        x[i][j].set(GRB_StringAttr_VarName, "x");
    }
}

//-----
//função objetivo
//-----
model.set(GRB_IntAttr_ModelSense, 1); //0 para maximizar ou 1 para

```

minimizar

```

model.update();                                     //atualiza o modelo

//-----
//restrição 1 :  $\sum_{i=1}^n x_{ij} = 1$ , forall  $j=1, \dots, n$ 
//-----
for (int j= 0; j< caminhao[0].nitens; j++)//para todo j
{
    GRBLinExpr r1= 0;//cria a expressão para a restrição 1
    for (int i= 0; i< caminhao[0].nitens; i++)//somatório em i
        r1+= x[i][j];
    model.addConstr(r1==1, "r1"); // adicionando restrição 1
}

//-----
//restrição 2 :  $\sum_{j=1}^n w_j x_{ij} \leq b y_i$ , forall  $i=1, \dots, n$ 
//-----
for (int i=0; i< caminhao[0].nitens; i++)//para todo i
{
    GRBLinExpr r2= 0;//cria a expressão para a restrição 2
    for (int j=0; j< caminhao[0].nitens; j++)//somatório em j
        r2+= peso[0][j]*x[i][j];
    model.addConstr(r2<= caminhao[0].capacidade*y[i], "r2"); //
adicionando restrição 2
}

//-----
//inicia a resolução do modelo
//-----

model.update();
model.write("Binpacking.lp");
model.optimize();

```

```

//-----
//Exporta a solução
//-----
FO= model.get(GRB_DoubleAttr_ObjVal);
for (int j= 0; j< caminhao[0].nitens; j++)
    respostay[j]= y[j].get(GRB_DoubleAttr_X);
for (int i= 0; i< caminhao[0].nitens; i++)
    for (int j= 0; j< caminhao[0].nitens; j++)
        respostax[i][j]= x[i][j].get(GRB_DoubleAttr_X);
printf("\n");
} catch(GRBException e)
{
    cout << "Gurobi - Código do erro = " << e.getErrorCode() << endl;
    cout << e.getMessage() << endl;
} catch(...)
{
    cout << "Gurobi - Erro durante otimização" << endl;
}
delete[] y;
for (int i= 0; i< caminhao[0].nitens; i++)
    delete[] x[i];
delete[] x;
delete env;//termina ambiente gurobi*/

time(&t_fim);
tempo= difftime(t_fim,t_ini);
}
//-----
//Impressão
//-----
//-----
//Função imprime: imprime os resultados no arquivo
//-----

```

```

void impressao()
{
    ofstream destino ("decbin11.txt");
    if (!destino)
        cerr << "\nErro ao abrir o arquivo bingurobi_solucao24\n\n";
    destino<<"\n-----\n";
    destino<<"Número do Problema\n";
    destino<<"-----\n";
    destino<<"nCódigo= "<<(caminhao[0].codigo+1);
    destino<<"\n-----\n";
    destino<<"Tempo\n";
    destino<<"-----\n";
    destino<<"O problema foi resolvido em "<<tempo<<" segundos.\n";
    destino<<"\n-----\n";
    destino<<"Comparação da FO encontrada\n";
    destino<<"-----\n";
    destino<<"\nFO= "<<FO;
    destino<<"\nPrimeira solução conhecida="<<caminhao[0].solucao;
    if( FO == caminhao[0].solucao)
        destino<<"\nA solução encontrada é equivalente a primeira solução obtida";
    else
        destino<<"\nA solução encontrada não equivale a primeira solução obtida";
    destino<<"\n-----\n";
    destino<<"Alocação da Carga\n";
    destino<<"-----\n";
    for (int j= 0; j< caminhao[0].nitens; j++)
        if(respostay[j])
        {
            destino<<"Caminhao "<<j<<":\n";
            for (int i=0; i< caminhao[0].nitens; i++)
                if(respostax[j][i])
                    destino<<"caixa "<<i<<" com peso "<<peso[0][i]<<"; ";
            destino<<"\n\n";
        }
}

```

```

    }

    destino<<"\n-----\n";
    destino<<"Solução\n";
    destino<<"-----\n";
    destino<<"\nYi:\n";
    for (int j= 0; j< caminhao[0].nitens; j++)
        destino<<respostay[j]<<" ";
    destino<<"\nXij:\n";
    for (int i=0; i< caminhao[0].nitens; i++)
    {
        for (int j= 0; j< caminhao[0].nitens; j++)
            destino<<respostax[i][j]<<" ";
        destino<<endl;
    }
}

```