

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

ISABELLA LIMA DE SOUZA
RAUL FERREIRA DE ANDRADE FRANCO

**UM COMPARATIVO DE DUAS METAHEURÍSTICAS APLICADAS A
PROBLEMAS DE BIN PACKING BIDIMENSIONAL**

Ituiutaba - MG
2017

ISABELLA LIMA DE SOUZA
RAUL FERREIRA DE ANDRADE FRANCO

**UM COMPARATIVO DE DUAS METAHEURÍSTICAS APLICADAS A
PROBLEMAS DE BIN PACKING BIDIMENSIONAL**

Trabalho de Conclusão de Curso,
apresentado a Universidade Federal de
Uberlândia, como parte das exigências
para a obtenção do título de Bacharel em
Engenharia de Produção.

Orientador: Prof. Dr. Jorge von Atzingen
dos Reis

ISABELLA LIMA DE SOUZA
RAUL FERREIRA DE ANDRADE FRANCO

UM COMPARATIVO DE DUAS METAHEURÍSTICAS APLICADAS A PROBLEMAS DE BIN PACKING BIDIMENSIONAL

Trabalho de Conclusão de Curso,
apresentado a Universidade Federal de
Uberlândia, como parte das exigências
para a obtenção do título de Bacharel em
Engenharia de Produção.

BANCA EXAMINADORA

Prof. Dr. Jorge von Atzingen dos Reis
Universidade Federal de Uberlândia

Prof. Dr. Antonio Álvaro de Assis Moura
Universidade Federal de Uberlândia

Prof. Dr. Marcus Vinícius Ribeiro Machado
Universidade Federal de Uberlândia

LISTA DE FIGURAS

Figura 1 - Exemplo de um problema de <i>bin packing</i> bidimensional.....	11
Figura 2 - Tipos básicos de problemas de corte e empacotamento	14
Figura 3 - Tipos intermediários de problemas: maximização da saída.....	15
Figura 4 - Tipos intermediários de problemas: minimização da entrada.....	16
Figura 5 - Método <i>Bottom-Left</i>	17
Figura 6 - Cromossomo para uma população de 100 objetos	28
Figura 7 - Reprodução	29
Figura 8 - Visualização Gráfica dos <i>bins</i> 1 e 2 do problema M1a com os 20 primeiros objetos inseridos por sorteio e o próximo objeto ordenado decrescente por área	35
Figura 9 - Exemplo de rotação de um objeto	35
Figura 10 - Visualização Gráfica da solução do problema M1a do Algoritmo Genético utilizando solução inicial com retorno	37
Figura 11 - Visualização gráfica Algoritmo Genético utilizando solução inicial com retorno.....	58
Figura 12 - Visualização gráfica Busca Tabu utilizando solução inicial com retorno	59
Figura 13 - Visualização gráfica Algoritmo Genético utilizando solução inicial sem retorno.....	60
Figura 14 - Visualização gráfica Busca Tabu utilizando solução inicial sem retorno	61

LISTA DE TABELAS

Tabela 1 - Características das categorias de problemas M1, M2 e M3	22
Tabela 2 - <i>Bins</i> das categorias de problemas M1, M2 e M3	23
Tabela 3 - Resultados metaheurísticas com método de solução inicial com retorno.	33
Tabela 4 - Soluções Busca Tabu e seus respectivos parâmetros.....	38
Tabela 5 - Comparação dos resultados em decorrência da ordem de inserção dos objetos	40
Tabela 6 - Problema M1a para cada percentual de mutação e crossover.....	42
Tabela 7 - Percentuais de mutação e crossover para cada problema	43
Tabela 8 - Resultados das metaheurísticas com método de solução inicial sem retorno.....	44
Tabela 9 - Problema M1a para cada percentual de mutação e crossover.....	50
Tabela 10 - Problema M1b para cada percentual de mutação e crossover	50
Tabela 11 - Problema M1c para cada percentual de mutação e crossover.....	51
Tabela 12 - Problema M1d para cada percentual de mutação e crossover	51
Tabela 13 - Problema M1e para cada percentual de mutação e crossover	52
Tabela 14 - Problema M2a para cada percentual de mutação e crossover	52
Tabela 15 - Problema M2b para cada percentual de mutação e crossover	53
Tabela 16 - Problema M2c para cada percentual de mutação e crossover.....	53
Tabela 17 - Problema M2d para cada percentual de mutação e crossover	54
Tabela 18 - Problema M2e para cada percentual de mutação e crossover	54
Tabela 19 - Problema M3a para cada percentual de mutação e crossover	55
Tabela 20 - Problema M3b para cada percentual de mutação e crossover	55
Tabela 21 - Problema M3c para cada percentual de mutação e crossover.....	56
Tabela 22 - Problema M3d para cada percentual de mutação e crossover	56
Tabela 23 - Problema M3e para cada percentual de mutação e crossover	57

RESUMO

Corte e empacotamento são processos bastante utilizados pelas indústrias e são constantemente estudados pela Pesquisa Operacional, pois seus custos afetam significativamente no custo final do produto. A redução destes gastos é importante para a diminuição dos desperdícios e obtenção de maiores lucros. Porém, são considerados problemas de natureza combinatória *NP-hard*, inviáveis de serem solucionados por algoritmos exatos, pois necessitam um tempo computacional avançado. O objetivo deste trabalho é realizar um comparativo no desempenho das metaheurísticas Busca Tabu e Algoritmo Genético na resolução de problemas de *bin packing* bidimensionais com *bins* de múltiplos tamanhos, com o intuito de alcançar soluções viáveis que otimizem o número de *bins* utilizados em um tempo computacional reduzido. Códigos computacionais foram desenvolvidos para ambas as metaheurísticas para a resolução de 15 problemas de *bin packing* contidos na *OR-Library* com até 150 objetos de no máximo 30x30 unidades de área. A utilização única e exclusiva da heurística de encaixe *Bottom-Left* se mostrou mais eficiente que as metaheurísticas implementadas e a Busca Tabu levou vantagem nos resultados em relação ao Algoritmo Genético em 80% dos problemas resolvidos.

Palavras-chave: Metaheurística, Busca Tabu, Algoritmo Genético, *Bin Packing*

ABSTRACT

Cutting and Packing processes are widely used by industries and constantly being studied by the field of Operational Research as their costs significantly affect the final cost of a product. It is important to reduce these costs by managing the industrial processes in a way that industries can minimize their waste and increase the profits. However, cutting and packing problems are considered NP-hard combinatorial optimization problems, and they can not be solved by exact algorithms because it requires a wide variation in computational time. The objective of this study is to compare the results obtained by Tabu Search and Genetic Algorithm metaheuristics in solving 2D multiple bin size bin packing problems in order to achieve viable solutions that optimize the number of bins in a reduced computational time. The metaheuristics were coded and 15 bin packing problems instances from the OR-Library were solved in order to evaluate their performance. The problems have a maximum of 150 objects and the largest object has 30x30 of dimensions. The Bottom-Left heuristic proved to be more efficient than the metaheuristics, and the Tabu Search achieved better results than the Genetic Algorithm in 80% of the problems.

Keywords: Metaheuristics, Tabu Search, Genetic Algorithm, Bin packing

SUMÁRIO

1	Introdução.....	8
2	Referencial Teórico.....	10
2.1	<i>Bin packing</i> bidimensional.....	10
2.1.1	Definição.....	10
2.1.2	Classificação.....	11
2.2	Heurística de encaixe <i>Bottom-Left</i>	16
2.3	Metaheurísticas.....	17
2.3.1	Busca Tabu.....	18
2.3.2	Algoritmo Genético	20
3	Metodologia.....	22
3.1	Etapas iniciais	24
3.2	Busca Tabu.....	26
3.3	Algoritmo Genético.....	28
4	Resultados.....	33
4.1	Busca Tabu.....	38
4.2	Algoritmo Genético	41
4.3	Busca Tabu x Algoritmo Genético	44
5	Conclusão.....	46
	Referências Bibliográficas	48
	APÊNDICE A – Soluções encontradas para cada problema testando pares de percentagens no Algoritmo Genético	50
	APÊNDICE B – Visualizações gráficas do problema M1a para a Busca Tabu e o Algoritmo Genético utilizando as soluções iniciais com retorno e sem retorno.....	58

1 Introdução

O problema de *bin packing*, também conhecido como agrupamento de entregas ou empacotamento de objetos, é um problema de otimização combinatória, onde se busca minimizar o número de *bins* necessários para transportar uma quantidade pré-determinada de objetos de tamanhos variados. Este problema pode ser dividido em três tipos: *Bin packing* (Unidimensional), *Bin packing* 2D (Bidimensional) e *Bin packing* 3D (Tridimensional). O problema unidimensional busca a otimização de apenas um parâmetro do problema, como por exemplo, o peso de uma caixa que armazena determinado número de objetos. O problema bidimensional, por sua vez, implica na alteração de duas variáveis que produzem diferentes soluções, e ambas devem ser otimizadas simultaneamente. O problema tridimensional é uma extensão do problema bidimensional, porém, cada objeto possui três variáveis características, como comprimento, largura e altura.

Problemas de corte e empacotamento são problemas de *bin packing* bidimensionais clássicos da pesquisa operacional e têm recebido atenção significativa por parte da comunidade acadêmica nas últimas décadas. O problema de empacotamento é evidenciado quando há a necessidade de se alocar alguns ou a totalidade dos objetos ou itens em estruturas denominadas *bins*, ou seja, espaços bidimensionais determinados, respeitando que cada objeto deve estar totalmente dentro deste espaço e que não devem se sobrepor entre si. Ao contrário do empacotamento, problemas de corte consistem na retirada de objetos de um plano bidimensional de modo a otimizar o número de objetos cortados e reduzir as perdas decorrentes dos cortes.

Diversos processos industriais como corte de bobinas de papel, rolos de tecido, placas de madeira, vidro ou metal, e a disposição de produtos em espaços disponíveis de *pallets*, contêineres, caminhões, para o armazenamento e transporte dos mesmos, são exemplos práticos que ilustram a otimização por corte e empacotamento.

Com o intuito de resolver problemas relacionados ao *bin packing*, procura-se um método de solução eficiente que forneça soluções em tempos de processamento reduzidos. Metaheurísticas não garantem a obtenção da solução ótima, pois não analisam todas as combinações possíveis de solução do

problema, o que levaria a tempos computacionais elevados. Portanto elas atendem ao requisito de encontrar diversas soluções em tempos de processamento reduzidos. Por este motivo, este trabalho busca a solução de problemas de *bin packing* bidimensionais com *bins* de diferentes tamanhos através da utilização de duas metaheurísticas, Busca Tabu e Algoritmo Genético, além de uma comparação entre o desempenho de ambas com relação aos resultados obtidos e o tempo computacional necessário para a obtenção da solução.

A utilização de metaheurísticas se deve ao fato de que os problemas de corte e empacotamento são classificados como *NP-hard* que, segundo Reis e Cunha (2009), são exclusivamente resolvidos por métodos heurísticos por se tratarem de problemas que, com a utilização de dados reais, se tornam inviáveis de serem resolvidos com algoritmos exatos, pois o tempo computacional se expressa em uma função não polinomial.

A realização deste trabalho se faz importante para mostrar a comparação de duas metaheurísticas na solução de problemas de *bin packing* bidimensionais. Assim, será possível avaliar se a otimização do problema de *bin packing* através do uso de metaheurísticas é vantajosa para a redução de custos, minimizando o número de *bins* necessários para armazenar os objetos ou itens.

A seção 2 apresenta a revisão bibliográfica utilizada durante este trabalho. Na seção 3 encontra-se a metodologia do trabalho, onde são detalhados os métodos referentes às metaheurísticas utilizadas. Em seguida, na seção 4, são apresentados os resultados obtidos pelo código implementado de ambas as metaheurísticas, possibilitando uma análise de seu desempenho e uma comparação destes resultados com o limitante inferior da solução. A seção 5 traz as conclusões e considerações finais sobre o problema proposto. Para finalizar o trabalho, são listadas as referências bibliográficas utilizadas e após as referências, encontra-se os apêndices do trabalho.

2 Referencial Teórico

Esta seção traz conceitos teóricos discutidos no decorrer do trabalho divididos em três partes: *bin packing* bidimensional, heurística *Bottom-Left* utilizada para a geração da solução inicial e as metaheurísticas. Na primeira parte, pode-se encontrar a definição do problema de *bin packing* bidimensional e os critérios de classificação de problemas de corte e empacotamento. A heurística *Bottom-Left* e as metaheurísticas Busca Tabu e Algoritmo Genético são definidas, respectivamente, na segunda e terceira subseção.

2.1 *Bin packing* bidimensional

A definição do problema bidimensional de *bin packing* pode ser encontrado na seção 2.1.1 e os critérios de classificação de problemas de corte e empacotamento na seção 2.1.2.

2.1.1 Definição

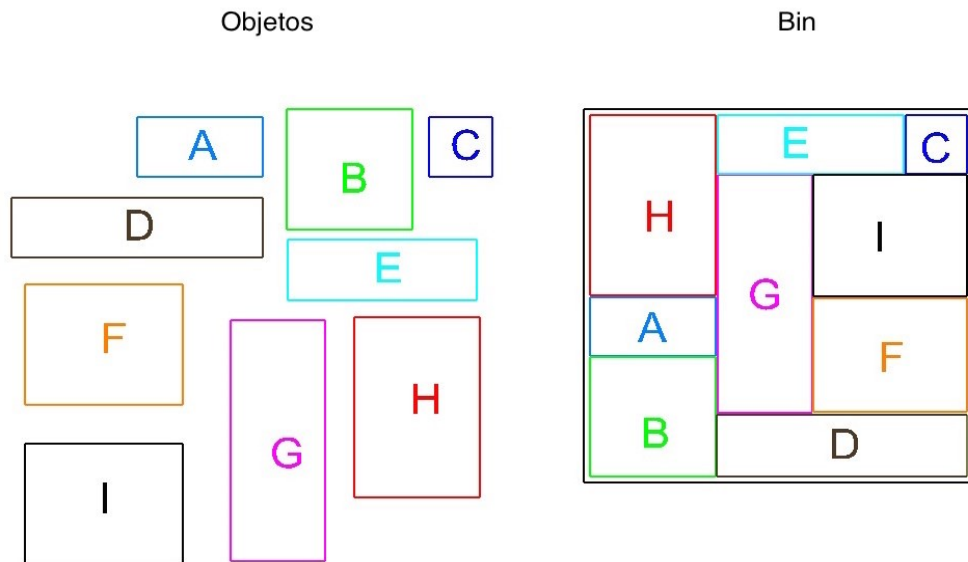
O modelo matemático para este problema pode ser definido segundo Pereira (2013) da seguinte forma:

No empacotamento com o objetivo de incluir todos os objetos na menor quantidade de *bins* possível, sendo os objetos retangulares pertencentes a um conjunto finito não nulo J com $j \in J = \{1, \dots, n\}$ e os *bins* retangulares a um conjunto finito não nulo I com $i \in I = \{1, \dots, m\}$ sendo (w_j, h_j) o comprimento e altura dos objetos e (W_i, H_i) dos *bins*.

De acordo com Mansano (2008) a natureza combinatória do processo de planejamento é onde se encontra as principais dificuldades na resolução do problema de *bin packing*, pois esse processo gera uma explosão de alternativas, principalmente em sistemas de médio e de grande porte.

A Figura 1 representa um exemplo de um problema de *bin packing* bidimensional, onde 9 itens são alocados em um *bin*.

Figura 1 - Exemplo de um problema de *bin packing* bidimensional



Fonte: Autoria Própria

As características do problema podem dificultar ainda mais sua resolução e consequentemente dar condições para o aparecimento de mais combinações de soluções. Como características que diferenciam esses problemas, podemos citar: objetos iguais ou com tamanhos relativamente diferentes, *bins* homogêneos ou heterogêneos, possibilidade ou não de rotação dos objetos para a obtenção de um melhor encaixe.

2.1.2 Classificação

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

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

Na maximização da saída, um conjunto de objetos tem que ser atribuído a um conjunto dado de *bins*. Porém, talvez nem usando

todo o conjunto de *bins* seja suficiente para acomodar todos os objetos. Portanto, o objetivo é encontrar a localização dos objetos sobre os *bins*, que maximize o benefício dos objetos atribuídos.

Consumidores realizam compras de produtos *on-line* pela rede de internet. Tais mercadorias devem ser submetidas a um sistema de distribuição para a entrega em diversas localidades. As entregas desses produtos geralmente ultrapassam a frota disponível para transporte, e, para isto, deve ser feita a maximização da saída dos produtos. Ou seja, é necessário maximizar o benefício de cada *bin* utilizado.

Na minimização da entrada, a totalidade dos objetos deve ser atribuída ao conjunto de *bins*. Ao contrário do anterior, o conjunto de *bins* é suficiente para acomodar todos os objetos. Portanto, o objetivo é encontrar a localização dos objetos sobre os *bins* que minimize o valor dos *bins* atribuídos.

O problema de minimização da entrada é bastante visto no âmbito logístico com a busca pela utilização do menor número de veículos para o transporte de um número específico de objetos.

- c) Sortimento dos objetos: Representa quão variados ou diversos são os objetos com relação as suas formas e medidas, mas levando em conta somente as dimensões geométricas relevantes. Os objetos podem ser agrupados em relativamente (com relação ao número total de objetos) poucas classes, nas quais, os objetos têm idênticas formas e dimensões. Por definição, se tratam como diferentes classes de objetos, se estes têm as mesmas formas e dimensões, mas suas orientações são diferentes. Para isto ocorre três casos distintos: Objetos idênticos, Sortimento fracamente heterogêneo e Sortimento fortemente heterogêneo.

Objetos idênticos se apresenta quando, todos os objetos têm tanto formas e dimensões iguais. Em casos onde o objetivo é a maximização da saída, assume-se que o único tipo de objeto demandado tem um número ilimitado de cópias ou exemplares.

Um sortimento fracamente heterogêneo se apresenta quando, a

demanda de cada tipo de objeto é relativamente grande, ou seja, existe um número alto de cópias ou exemplares por tipo de objeto, e este pode ser ou não restringido por um limite máximo.

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

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

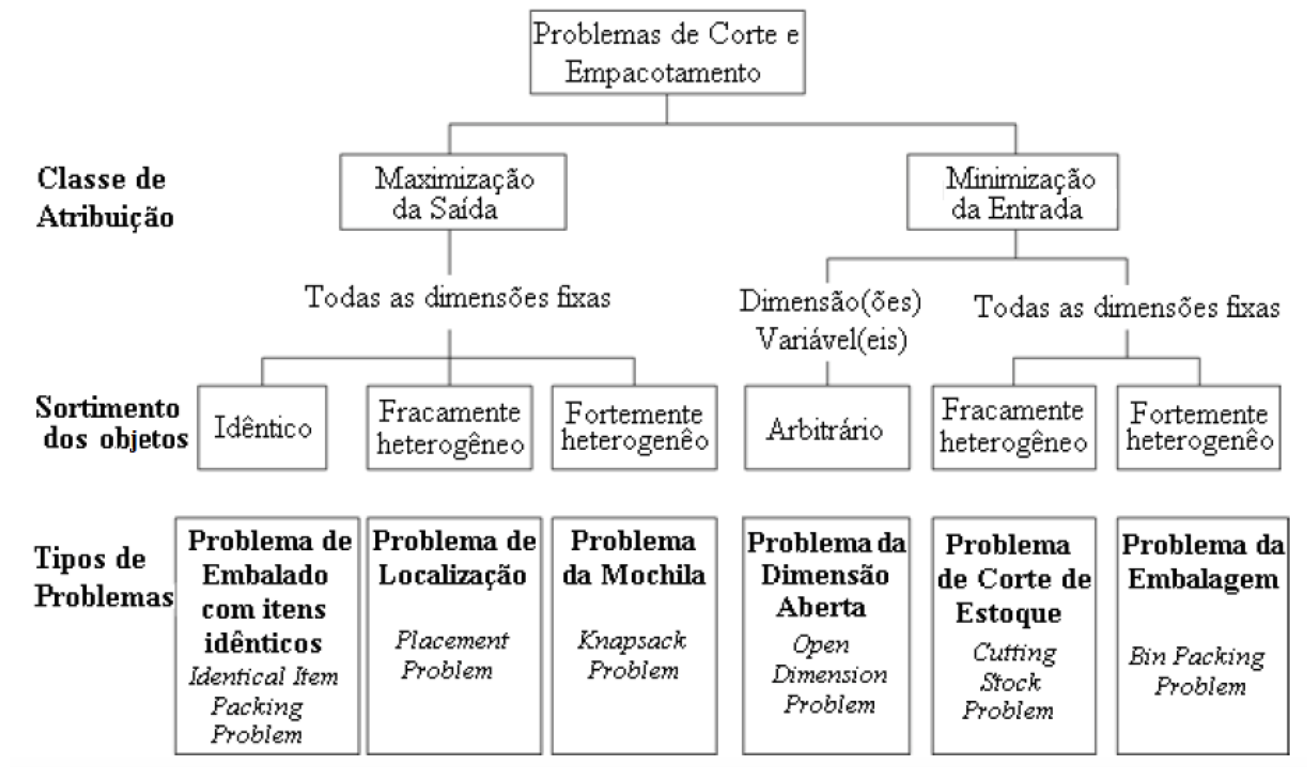
Quando existe só um *bin* pode ocorrer que este tenha todas suas dimensões fixas (sejam dadas) ou que uma ou mais de suas dimensões sejam variáveis ou ilimitadas.

Quando existem vários *bins*, cada um destes tem suas dimensões fixas, assim não existe o caso de vários *bins* com dimensões variáveis. Deste modo, a existência de vários *bins* significa que, igualmente aos objetos, estes possam ser classificados novamente nas três possíveis categorias de sortimento.

- e) Forma dos objetos: representa geometricamente como estão definidos os objetos, nos casos aonde são relevantes e necessárias duas ou três dimensões geométricas para representar os objetos. Podem-se identificar duas classes de objetos: Os regulares (retângulos, círculos, paralelepípedos, cilindros, bolas, etc.) e os irregulares.

A Figura 2 mostra as nomenclaturas de tipos básicos de problemas de corte e empacotamento de acordo com a classe de atribuição, as dimensões e o sortimento dos objetos e *bins* para cada problema.

Figura 2 - Tipos básicos de problemas de corte e empacotamento



Fonte: Martínez (2014), modificado pelo autor

O tipo básico dos problemas a serem resolvidos neste trabalho se encaixa no tipo de Problema de Embalagem, em que todas as dimensões dos objetos e dos *bins* são fixas. A classe de atribuição é a minimização da entrada, pois a totalidade dos objetos devem ser atribuídos aos *bins* e a totalidade dos *bins* tem capacidade suficiente para tal armazenamento.

Na Figura 3 tem-se o panorama de tipos intermediários de problemas de corte e empacotamento derivados dos tipos básicos de problemas com a maximização da saída como classe de atribuição. As características dos *bins* e o sortimento dos objetos são as características que diferenciam tais problemas.

Figura 3 - Tipos intermediários de problemas: maximização da saída

Sortimento dos objetos Características dos bins		Idênticos	Fracamente Heterogêneos	Fortemente Heterogêneos
Todas as dimensões fixas	Um único bin	Problema de acondicionamento de itens idênticos <i>Identical Item Packing Problem</i> IIPP	Problema da localização num único bin <i>Single Large Bin Placement Problem</i> SLBPP	Problema de uma única mochila <i>Single Knapsack Problem</i> SKP
	Bins idênticos	X	Problema da localização em múltiplos bins idênticos <i>Multiple Identical Large Bin Placement Problem</i> MILBPP	Problema das múltiplas mochilas idênticas <i>Multiple Identical Knapsack Problem</i> MIKP
	Bins Heterogêneos		Problema da localização em múltiplos bins heterogêneos <i>Multiple Heterogeneous Large Bin Placement Problem</i> MILBPP	Problema das múltiplas mochilas heterogêneas <i>Multiple Heterogeneous Knapsack Problem</i> MIKP

Fonte: Martínez (2014), modificado pelo autor

A minimização da entrada como classe de atribuição para problemas de corte e empacotamento tem seu panorama de tipos intermediários de problemas descrito na Figura 4.

Figura 4 - Tipos intermediários de problemas: minimização da entrada

Sortimento dos objetos Características dos bins		Fracamente Heterogêneos	Fortemente Heterogêneos
Todas as dimensões fixas	Bins idênticos	Problema de corte de estoque em bins de tamanhos idênticos <i>Single Stock Size</i> <i>Cutting Stock Problem</i> SSSCSP	Problema da embalagem em bins de um único tamanho <i>Single Bin Size</i> <i>Bin Packing Problem</i> SBSBPP
	Bins fracamente heterogêneos	Problema de corte de estoque múltiplos bins <i>Multiple Stock Size</i> <i>Cutting Stock Problem</i> MSSCSP	Problema da embalagem em bins múltiplos tamanhos <i>Multiple Bin Size</i> <i>Bin Packing Problem</i> MBSBPP
	Bins fortemente heterogêneos	Problema residual de corte de estoque <i>Residual</i> <i>Cutting Stock Problem</i> RCSP	Problema residual da embalagem <i>Residual</i> <i>Bin Packing Problem</i> RBPP
Um único bin com dimensões variáveis		Problema de dimensão aberta <i>Open Dimension Problem</i> ODP	

Fonte: Martínez (2014), modificado pelo autor

2.2 Heurística de encaixe *Bottom-Left*

Outro fator que aumenta a dificuldade do problema de *bin packing* bidimensional é a necessidade de não somente selecionar quais objetos estão em cada *bin*, mas garantir que os referidos objetos realmente caibam dentro das dimensões do *bin* sem extrapolar seus limites e sem haver sobreposição entre os objetos.

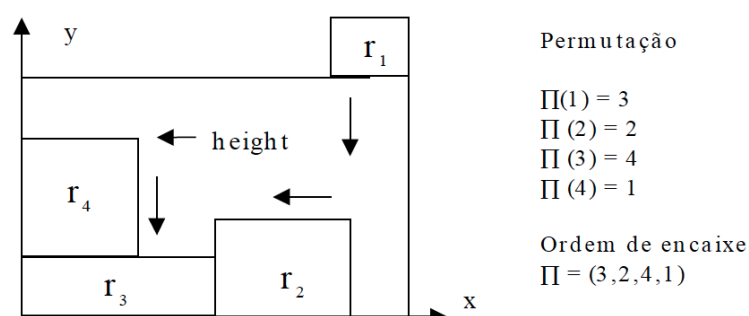
Por exemplo, se três objetos de peso 60kg cada precisam ser transportados em *bins* de capacidade máxima de 100kg, o limitante inferior da quantidade de *bins* necessários é a soma dos pesos dos objetos dividido pela capacidade máxima dos *bins*. Ou seja, seria necessário aproximadamente dois *bins*, com capacidade total de 200kg, para transportar 180kg de objetos. Entretanto esta condição não é válida pois, para a utilização de apenas dois *bins*, um deles deveria conter dois objetos que somados ultrapassariam a capacidade máxima do *bin*.

No âmbito bidimensional acontece da mesma forma. A alocação de dois objetos com dimensões de 10x10 e 5x4 unidades de altura e comprimento (120 unidades de área) em um *bin* de 10x12 (120 unidades de área) não é válida. A soma das áreas dos objetos não ultrapassa a área total do *bin*, porém, quando a tentativa de encaixe é realizada, nota-se que os objetos não podem ser encaixados naquele *bin* pois, diferente do esperado, ultrapassam as dimensões do mesmo.

Para resolver esse problema e atuar de forma correta, deve-se usar uma estratégia para encontrar os espaços vazios no *bin* e verificar se o objeto que está em estudo para ser alocado se adequa às dimensões destes espaços.

Segundo Constantino e Júnior (2002), o *Bottom-Left* segue duas prioridades para o encaixe de objetos em *bin* retangular. Primeiramente, insere o objeto o mais próximo possível do eixo x e, então, o mais próximo do eixo y do *bin*. Portanto, o processo de encaixe se inicia com o primeiro objeto no canto inferior esquerdo e tem os próximos objetos encaixados seguindo estes dois critérios. Assim que todos os objetos são encaixados é possível determinar o desperdício total do *bin*. Um exemplo da técnica *Bottom-Left* pode ser visto na Figura 5. A lista de objetos segue a ordem de encaixe: 3, 2, 4, e 1.

Figura 5 - Método *Bottom-Left*



Fonte: Constantino, Júnior (2002).

2.3 Metaheurísticas

Segundo Sucupira (2004), uma metaheurística define métodos heurísticos com aplicações em diferentes problemas de otimização, facilmente modificados para adaptação a problemas específicos. *Simulated Annealing*,

Busca Tabu, *Iterated Local Search*, algoritmos evolutivos e *Ant Colony* são exemplos de metaheurísticas.

As heurísticas são um conjunto de regras pré-definidas para a resolução de um problema previamente conhecido. As metaheurísticas são heurísticas que possuem uma estrutura de refinamento das soluções. As metaheurísticas são combinações dos métodos heurísticos (MANSANO, 2008). Essas estratégias apresentam um conjunto de técnicas de otimização adaptadas para lidar com problemas complexos e que apresentam característica da explosão combinatória.

2.3.1 Busca Tabu

A Busca Tabu foi um método sugerido por Fred Glover na década de 80. Esse método foi extremamente difundido para a resolução de problemas complexos de diferentes segmentos de pesquisa operacional.

De acordo com HIGGINS (2001), a Busca Tabu é baseada em um estabelecimento de movimentos para transformar uma solução atual em uma solução vizinha. Movimentos recentes são inseridos na Lista Tabu para que sua movimentação seja proibida enquanto permanecer na lista. A cada iteração da metaheurística Busca Tabu, procura-se uma solução de vizinhança que é gerada por movimentos não proibidos. Estes movimentos se tornam permitidos quando a solução gerada satisfaz um critério de aspiração.

Sabe-se que o método Busca Tabu tem a solução inicial s_0 como ponto de partida. Modificações são realizadas a cada iteração, gerando soluções que estão em um subconjunto V de vizinhança dentre um conjunto $N(s)$ de possíveis soluções. Nomeia-se estas soluções derivadas da solução s de s' , e a melhor s' encontrada é utilizada na próxima iteração para geração de um novo subconjunto de vizinhança (SUBRAMANIAN, 2011).

Segundo Subramanian (2011), a maioria dos métodos de busca mantém arquivada a melhor solução encontrada dentre as iterações realizadas, ou seja, o resultado da $f(s^*)$ de uma solução s^* . A Busca Tabu se difere destes métodos ao arquivar não apenas os dados da solução s^* , mas também um conjunto das últimas soluções encontradas. Este fato garante a criação da Lista Tabu,

utilizada como critério de movimentação na criação do subconjunto V de vizinhança, impedindo soluções repetidas em iterações próximas.

A Lista Tabu tem a finalidade de gerar uma variabilidade de soluções, para que em iterações próximas não haja repetição de soluções geradas pela mesma movimentação. Os movimentos repetidos se encontram, portanto, em um conjunto T de iterações recentes. O critério para retirada destes movimentos da Lista Tabu segue uma lógica FIFO (First In First Out). Assim, o primeiro objeto inserido é o primeiro a ser retirado após a lista atingir o seu tamanho máximo (SUBRAMANIAN, 2011).

O método parte de uma solução inicialmente conhecida e sua nomeação Busca Tabu provém da finalidade da metaheurística em definir restrições na busca de uma solução vizinha, determinadas como proibições de movimentos.

Para projetar um algoritmo Busca Tabu é necessário especificar os componentes:

- a) Critério de escolha da solução vizinha seguinte;
- b) Seleção dos atributos do movimento;
- c) Memória de curto prazo para armazenar a lista tabu;
- d) Tamanho da lista tabu (número de iterações);
- e) Critério de aspiração.

O critério de escolha da solução vizinha seguinte trata de definir qual a condição para que seja realizada a movimentação para soluções diferentes da solução atual (soluções vizinhas). A seleção dos atributos do movimento traz a escolha de quais parâmetros serão analisados para que possa ser feita a mudança (o peso ou as dimensões dos objetos ou itens, por exemplo). Deve se criar uma estrutura de armazenamento de curto prazo das soluções encontradas no decorrer do problema denominada Lista Tabu. A definição do tamanho da lista também é importante, pois quanto maior o tamanho da lista, mais movimentações serão armazenadas na mesma e consequentemente proibidas de ser realizadas, dando opções de busca de outros espaços de soluções. E após algumas mudanças nas soluções, pode ocorrer de que uma movimentação que foi analisada como sendo a melhor para aquela iteração esteja contida na lista tabu, mas caso ela seja realizada, ela trará uma solução corrente melhor do que as já encontradas naquele momento. Nesse caso, tem se o critério de

aspiração, que nada mais é do que uma alternativa onde se o movimento está proibido, mas vai gerar uma solução melhor, ele será liberado para ser realizado, mas somente nessas condições.

Em Mansano (2008), a metaheurística Busca Tabu foi utilizada para a resolução de um problema de PPET (Problema do Planejamento da Expansão dos Sistemas de Transmissão de Energia Elétrica) que consiste em determinar quando, onde e que tipos de linhas e/ou transformadores devem ser instalados na rede para que o sistema opere de forma adequada para uma demanda futura predeterminada e investindo o mínimo possível. A técnica Busca Tabu mostrou-se eficiente em encontrar valores ótimos para sistemas de pequeno e médio porte, e boas soluções para sistemas de grande porte. Em Subramanian (2011), a Busca Tabu foi utilizada na resolução do problema de PAAS (Problema de Alocação de Aulas à Salas) que se trata de alocar aulas, com horários de início e término previamente programados, a um número fixo de salas. A Busca Tabu produziu soluções finais de qualidade compatível com os valores dos pesos considerados, mostrando-se adequada.

2.3.2 Algoritmo Genético

Segundo Lucas (2002), na busca pela otimização os algoritmos genéticos (AG) se inspiram no princípio da seleção natural, proposto por Darwin em 1858. Os algoritmos genéticos submetem uma população com possíveis respostas do problema a ser resolvido através deste princípio de evolução das espécies.

Segundo Sridhar et al. (2017), as etapas do algoritmo genético são: (i) Geração da População; (ii) Reprodução (Crossover); (iii) Mutação; (iv) Seleção; (v) Inversão; e (vi) Parada.

A primeira etapa para a utilização do Algoritmo Genético é a geração da população inicial aleatoriamente, baseando-se em lógicas de probabilidade e no fato que cada indivíduo é composto por cromossomos na biologia. O tamanho da população depende da natureza do problema, mas geralmente engloba milhares de soluções possíveis.

Cromossomos são providenciados através do Crossover, que é a segunda etapa do Algoritmo Genético, onde os pais se reproduzem para formar

a melhor solução possível. Isto permite que os filhos gerados tenham as características genéticas de ambos os pais.

De acordo com Sridhar et al. (2017), a função objetivo mede a qualidade da solução gerada. Uma vez definida, o Algoritmo Genético aplica o Crossover, Mutação, Inversão e Seleção repetidamente na busca da solução ótima. A função objetivo é uma função de maximização ou minimização utilizada para calcular o valor relativo entre a solução obtida e a esperada.

Algumas vezes os valores das funções objetivos obtidos pelos filhos tendem a se manter próximas ao valor do ponto ótimo, pois cadeias de cromossomos tendem a repetir dentre os filhos gerados, e isso leva a estagnação dos resultados. A mutação previne esta condição de estagnação. Nela, novas soluções são geradas através de pequenas mudanças em prévias soluções.

A etapa da Seleção é de grande importância na redução do tempo computacional de execução do AG. A seleção ideal garante uma rápida convergência dos resultados, e pode ser um processo aleatório ou sequencial.

É comum que depois de várias iterações, os inícios das cadeias de cromossomos dos pais permaneçam os mesmos, ainda que depois de várias gerações, e podem gerar a estagnação dos resultados. A inversão é a quinta etapa do AG, onde se é estabelecido um ponto único e aleatório de troca.

O limite no número de gerações ou o alcance de um valor satisfatório da função objetivo são condições de parada comumente utilizadas para interromper a execução do Algoritmo Genético. O critério ainda pode ser um número fixo de iterações que não produzem melhores resultados, tempo de execução ou custos atingidos. Este critério de parada compõe a última etapa da implementação do Algoritmo Genético.

3 Metodologia

Os objetos dos problemas resolvidos neste trabalho são fortemente heterogêneos e os *bins* são fracamente heterogêneos, ou seja, os objetos possuem grande variação de dimensões e os *bins* possuem dimensões diferentes, porém levemente similares. Portanto, se referem a problemas de embalagem em *bins* de múltiplos tamanhos.

Os métodos Busca Tabu e Algoritmo Genético foram ambos desenvolvidos através de códigos C/C++.

Os dados de entrada do problema de *bin packing* a ser resolvido estão disponibilizados em OR-Library (2017). O arquivo utilizado é o “binpacktwo.xls” que contém diversos problemas a serem estudados com todas as informações necessárias para a resolução dos mesmos.

Para este trabalho foram resolvidos os problemas de categorias M1, M2 e M3, que se distinguem entre si pelo número de objetos, número de *bins* disponíveis e pelas dimensões dos mesmos e em cada categoria encontram-se 5 problemas nomeados de a até e. A Tabela 1 lista o número de objetos, número de *bins* e a dimensão do maior objeto para cada categoria de problema.

Tabela 1 - Características das categorias de problemas M1, M2 e M3

Categoria	Número de Objetos	Número de <i>bins</i>	Dimensão do maior objeto
M1	100	16	9x9
M2	100	18	28x28
M3	150	20	30x30

Fonte: OR-Library (2017)

Devido a diferenciação nas dimensões dos objetos e ao número de *bins* disponíveis, as categorias M1, M2 e M3 podem ser classificadas pelo nível de dificuldade de resolução pelos códigos computacionais em:

- a) Categoria M1 – problemas de nível fácil, com rápida resolução;
- b) Categoria M2 – problemas de nível intermediário;
- c) Categoria M3 – problemas difíceis, com resolução demorada.

A Tabela 2 mostra as dimensões dos *bins* disponíveis para cada grupo de problemas.

Tabela 2 - *Bins* das categorias de problemas M1, M2 e M3

Categoria	Comprimento do <i>bin</i>	Altura do <i>bin</i>	Área	Quantidade de <i>bins</i>
M1	20	20	400	2
	10	30	300	3
	10	10	100	3
	20	30	600	2
	10	20	200	3
	10	40	400	3
M2	60	40	2400	2
	30	90	2700	3
	30	30	900	4
	60	90	5400	2
	30	60	1800	4
	30	120	3600	3
M3	60	50	3000	3
	30	100	3000	3
	40	30	1200	3
	60	100	6000	4
	30	60	1800	4
	30	120	3600	3

Fonte: OR-Library (2017)

3.1 Etapas iniciais

A fase inicial da execução dos códigos desempenhada pelas metaheurísticas Busca Tabu e Algoritmo Genético se divide em leitura de dados e preparação para geração da solução inicial.

Os dados do arquivo “binpacktwo.xls” são transferidos para um arquivo de texto para que o código faça a leitura e armazenamento dos dados. Os dados de entrada encontrados nos arquivos contêm:

- a) Nome do problema - Utilizado para identificação do problema;
- b) Número de *bins* disponíveis e suas respectivas dimensões (dados disponíveis na Tabela 2). A ordem decrescente de área é o critério para disposição inicial dos *bins* nos arquivos de entrada, garantindo que tenham prioridade de utilização. Ou seja, os *bins* são organizados em ordem decrescente de área para que as metaheurísticas tenham prioridade inicial de escolha daqueles que possuem maior área, uma vez que, geralmente, possuem maior capacidade de armazenagem de objetos e facilitam a redução do número de *bins* final;
- c) Número de objetos a serem alocados e suas respectivas dimensões.

Após a leitura de dados, as metaheurísticas se preparam para a criação da solução inicial que é o ponto de partida para a geração de soluções vizinhas. Esta preparação é caracterizada pela criação de matrizes que representam cada um dos *bins* disponíveis, compostas por linhas e colunas com dimensões de 1×1 unidade de área. Estas matrizes permitem uma visualização gráfica de como os objetos são encaixados em cada um dos *bins*. Em uma placa retangular com dimensões de $m \times n$ unidade de área, por exemplo, onde m a altura e n é o comprimento do *bin*, tem-se a matriz $A = (a_{i,j})_{m \times n}$ em que $i \in I = \{1, \dots, m\}$ e $j \in J = \{1, \dots, n\}$.

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \dots & a_{2,n} \\ a_{3,1} & a_{3,2} & a_{3,3} & \dots & a_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & a_{m,3} & \dots & a_{m,n} \end{bmatrix}$$

A matriz foi espelhada verticalmente com a finalidade de proporcionar melhor visualização dos *bins* através do método *Bottom-Left*, que inicia seu encaixe pelo nível inferior esquerdo.

$$\begin{bmatrix} a_{m,1} & a_{m,2} & a_{m,3} & \dots & a_{m,n} \\ \vdots & \vdots & \vdots & & \vdots \\ a_{3,1} & a_{3,2} & a_{3,3} & \dots & a_{3,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \dots & a_{2,n} \\ a_{1,1} & a_{1,2} & a_{1,3} & \dots & a_{1,n} \end{bmatrix}$$

As metaheurísticas Busca Tabu e Algoritmo Genético apresentam suas particularidades quanto à ordem de inserção dos objetos e à escolha dos *bins* para a movimentação, pois enquanto na Busca Tabu os objetos são sequenciados em ordem decrescente de área e alocados um a um seguindo essa sequência, no Algoritmo Genético uma porcentagem é selecionada aleatoriamente para diferenciar os indivíduos da solução, representando os primeiros objetos a serem inseridos nos *bins*. Após o armazenamento destes objetos, os objetos seguintes também utilizam a ordenação decrescente usada pela Busca Tabu. Com estas particularidades definidas, a movimentação de objetos segue a lógica do método *Bottom-Left*, onde cada objeto procura espaço disponível na matriz A desde sua base $a_{1,1}$ até $a_{m,n} \forall i$, onde $j = 1, 2, \dots, n$.

O teste de movimentação do objeto é realizado para todo espaço livre encontrado levando em consideração o encaixe horizontal ou vertical do objeto. Se uma matriz é totalmente percorrida e a movimentação do objeto não é realizada, é possível concluir que este *bin* não possui espaço suficiente para o encaixe deste objeto. Este trabalho utiliza duas possibilidades de ação quando este fato acontece, que serão chamados de: solução inicial sem retorno e solução inicial com retorno.

Na solução inicial sem retorno, após um objeto não encontrar espaço suficiente em um *bin*, o teste de movimentação deste objeto e de todos os seguintes é feito a partir do próximo *bin* disponível em diante, ou seja, o código não revisitará aqueles *bins* que já foram considerados incapazes de armazenar algum objeto. Na solução inicial com retorno, visando uma análise mais criteriosa, a movimentação de cada objeto é testada desde o primeiro *bin*

seguindo a ordem que foram lidos no arquivo de entrada. Assim, o código analisa todos os espaços disponíveis até que o encaixe seja possível.

3.2 Busca Tabu

A criação da solução inicial utiliza uma estratégia de ordenação dos objetos em ordem decrescente de área utilizando o algoritmo de ordenação *Quick Sort*, para então colocá-los um a um nos *bins* que possuem capacidade suficiente. A solução é definida como solução estrela que representa a melhor solução encontrada no estado atual e mantida salva para comparações com futuras soluções. O valor das soluções é calculado através da função objetivo caracterizada pela seguinte equação:

$$FO = \text{Número de bins} + \frac{\text{Área ocupada do bin mais vazio}}{\text{Área do bin}}$$

O objetivo principal é minimizar o número de *bins* necessários, e para melhor diferenciação dos resultados da equação, soma-se a este valor a razão que determina a porcentagem de ocupação do *bin* mais vazio. Se esta razão é minimizada, tem-se que esta solução está mais perto de eliminar um *bin*.

O primeiro passo da Busca Tabu é a determinação do critério de escolha da solução vizinha seguinte, ou seja, a condição analisada para gerar novas soluções no espaço de busca de soluções local. O critério escolhido é a busca pelo *bin* que possui maior folga, e caso haja empate, prioriza-se aquele com menor número de objetos pela maior facilidade de eliminação. Seu objeto de menor área é selecionado para movimentação a outro *bin* que tenha capacidade de recebê-lo, pois, por possuir menor área, tem maior facilidade de alocação. Caso seja encontrado algum *bin*, o movimento é então realizado.

Se na tentativa de movimentação de um objeto não seja encontrado nenhum *bin* que tenha capacidade para recebê-lo, seleciona-se outro *bin* e repete o processo de busca do menor objeto dentro dele para tentar realizar a movimentação. Com a movimentação seja concretizada, tem-se então uma nova

7. $T \leftarrow \emptyset$; {Lista Tabu}
8. Enquanto (Iteracao – Melhoriteracao < BT_{max}) faça
9. Iteracao $\leftarrow$ Iteracao + 1;
10. Seja $s' \leftarrow s \oplus m$ o melhor elemento de V (conjunto de movimentações) $\subseteq N(s)$ (espaço de soluções vizinhas) tal que o movimento m não seja tabu ($m \notin T$) e que seja válido pela heurística de encaixe *Bottom-Left* ou se tabu, atenda ao critério de aspiração, isto é, número de objetos no bin igual a 1, ou seja, $f(s') < f(s^*)$
11. Atualize a Lista Tabu;
12. $s \leftarrow s'$;
13. Se $f(s') < f(s^*)$ então
14. $s^* \leftarrow s'$;
15. Melhoriteracao $\leftarrow$ Melhoriteracao + 1;
16. Fim-se;
17. Fim-enquanto;
18. Retorne s^* ;

Fim BT

3.3 Algoritmo Genético

Diferentemente da metodologia de execução da metaheurística Busca Tabu, o Algoritmo Genético baseia-se em um conjunto de possíveis soluções (indivíduos) e não em uma solução única. A este conjunto de soluções dá-se o nome de população. Cada indivíduo da população possui um cromossomo, que, no problema a ser resolvido, representa a ordem de encaixe dos objetos nos *bins* armazenada em um vetor. Os objetos, então, representam os genes do cromossomo.

Figura 6 - Cromossomo para uma população de 100 objetos

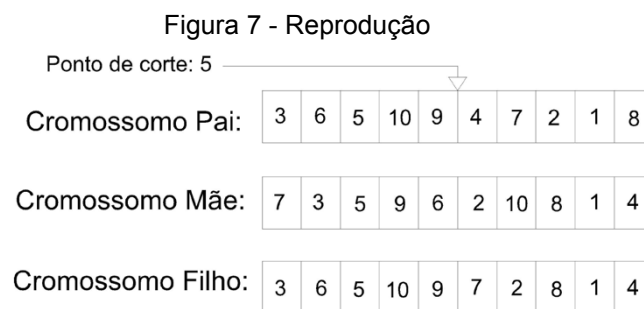
...	45	89	11	52	100	26	70	33	...
-----	----	----	----	----	-----	----	----	----	-----

Fonte: Autoria Própria

Novas populações são geradas através de modificações na população inicial, mantendo sempre o melhor indivíduo gerado como solução, até que o critério de parada seja atingido. O tamanho da população é de 50 indivíduos

devido a limitação computacional apresentada na resolução dos problemas. Segue-se a execução das 6 etapas da metaheurística:

- a) Geração da População – os 50 indivíduos iniciais são gerados. Com o propósito de diferenciá-los, 20% dos objetos são escolhidos aleatoriamente e inseridos no cromossomo do indivíduo. Os 80% dos objetos restantes têm suas áreas alinhadas de forma decrescente e são inseridos ao final do vetor, finalizando assim a lista com a ordem de alocação de todos os objetos.
- Uma vez que o indivíduo tem seu cromossomo finalizado, os objetos são encaixados um por um através da heurística *Bottom-Left* nos *bins* disponíveis, que são previamente organizados com área decrescente no arquivo de entrada. Após concluir a lista, têm-se a solução inicial.
- b) Reprodução (*Crossover*) - novas soluções são geradas através do cruzamento de todos os pares possíveis de indivíduos da solução inicial. Deste cruzamento surgem 1225 soluções filhas ou indivíduos filhos, que provêm de 50% da ordem de encaixe dos objetos do indivíduo-pai e 50% do indivíduo-mãe. A Figura 7 exemplifica o processo de reprodução para um indivíduo com 10 objetos.



Fonte: Autoria Própria

O cromossomo filho se inicia com os genes do cromossomo pai até o ponto de corte. Depois, percorre o cromossomo mãe e utiliza aqueles que não estavam presentes na parte do cromossomo pai selecionada.

- c) Mutação – 50 novas soluções são geradas através da mutação dos indivíduos da solução inicial. Esta mutação ocorre de forma semelhante à movimentação estabelecida na metaheurística Busca Tabu. Inicia-se uma busca pelo objeto de menor área no *bin* que possui maior folga de capacidade, e, caso haja vários *bins* com a mesma folga, escolhe-se aquele que possui o menor número de

objetos. Este objeto realiza então testes de movimentação desde o *bin* que tem a capacidade disponível mais próxima a sua área até o último *bin* da lista, já que estão organizados de forma crescente, o que aumenta a probabilidade de eliminação daqueles que possuem poucos objetos e utiliza a capacidade disponível de um *bin* quase cheio.

- d) Seleção - Nesta etapa, 50 indivíduos gerados pela reprodução e mutação serão selecionados para formarem a próxima geração. O processo de algoritmo genético lida com um grande número de indivíduos e, para selecioná-los, se faz necessário o uso de um parâmetro de comparação entre as soluções. São calculados os resultados da função objetivo para cada um dos indivíduos mutados e cruzados que segue a mesma equação utilizada pela metaheurística Busca Tabu.

O segundo fator da equação é importante na diferenciação dos indivíduos, uma vez que muitos deles, principalmente aqueles provenientes do Crossover, possuem a mesma quantidade de *bins* utilizados. Este fator permite a priorização da sobrevivência de indivíduos que têm maior probabilidade de eliminação de um *bin* ao decorrer das gerações.

As probabilidades de ocorrência de reprodução e mutação são estabelecidas como parâmetro para a metaheurística Algoritmo Genético. Para o desenvolvimento deste trabalho, cada problema utiliza a melhor combinação de porcentagens entre o grupo de indivíduos cruzados e mutados.

- e) Inversão - O ponto de troca definido para a etapa de inversão é a troca entre indivíduos selecionados e não selecionados na etapa anterior. A etapa de inversão aponta soluções que possuem os piores valores de FO e os insere no lugar de indivíduos escolhidos aleatoriamente dentre os selecionados para a próxima geração. Ou seja, indivíduos aleatórios são mortos para a sobrevivência de indivíduos ruins. Este evento representa a variabilidade genética da população e cada problema utiliza um percentual de 60% de troca, ou seja, 30 indivíduos.
- f) Parada - o critério de parada utilizado é um determinado número de gerações. Os 15 problemas serão resolvidos com o parâmetro de 100 gerações para estabelecer a geração limite de estagnação dos resultados, e este valor será fixado para realização dos outros testes.

Segue o pseudocódigo do algoritmo genético:

Início AG

1. Definir os parâmetros: p_{mut} (probabilidade de mutação), p_{cross} (probabilidade de crossover) e r (número de populações geradas sem melhora como critério de parada).
2. Seja $P = [A_1, A_2, \dots, A_m]$ a solução corrente, $P_0 = [X_1, X_2, \dots, X_m]$ a população inicial, $P_{mut} = [Y_1, Y_2, \dots, Y_m]$ a população inicial modificada por mutação, $P_{cross} = [Z_1, Z_2, \dots, Z_{n_{filhos}}]$ a população inicial modificada por crossover, $N = [W_1, W_2, \dots, W_o]$ os indivíduos não selecionados na etapa de Seleção, $P_{pior} = [K_1, K_2, \dots, K_{n_{inversao}}]$ os piores indivíduos onde $P_{pior} \subset N_{seleção}$ e $n_{inversao}$ é o número de indivíduos selecionados para a etapa de Inversão, I^* o melhor indivíduo encontrado e k o número de populações geradas sem melhora de X^* .
3. $P \leftarrow P_0$ {Solução Corrente}
4. $I^* \leftarrow I$ {Melhor indivíduo}
5. $k \leftarrow 1$
6. Enquanto ($k \leq r$) faça {Critério de parada}
7. Para $i \leftarrow 1$ até m faça {Mutação}
8. Realizar função: $mutacao(X_i)$
9. $X_i \in P_{mut}$
10. Calcular $f_{mut}(i)$
11. Fim-para
12. Para $filho \leftarrow 1$ até n_{filhos} faça {Crossover}
13. Escolher pai X_{pai}
14. Escolher pai $X_{mãe}$
15. Realizar função: $crossover(X_{pai}, X_{mãe})$
16. $X_{filho} \in P_{cross}$
17. Calcular $f_{cross}(filho)$
18. Fim-para
19. Para $i \leftarrow 1$ até $(m \times p_{mut})$ faça {Seleção}
20. Para $j \leftarrow 1$ até m
21. Para melhor $f_{mut}(j), Y_j \in P$
22. Fim-para
23. Fim-para
24. Para $i \leftarrow 1$ até $(m \times p_{cross})$ faça
25. Para $j \leftarrow 1$ até n_{filhos}
26. Para melhor $f_{cross}(j), Z_j \in P$
27. Fim-para
28. Fim-para
29. Para $i \leftarrow 1$ até m faça {Inversão}
30. Se $(Y_i \in P_{mut})$ e $(Y_i \notin P)$ então

```

31.           $Y_i \in N_{seleção}$ 
32.      Fim-se
33.  Fim-para
34.  Para  $i \leftarrow 1$  até  $n_{filhos}$ 
35.      Se  $(Z_i \in P_{cross})$  e  $(Z_i \notin P)$  então
36.           $Z_i \in N_{seleção}$ 
37.      Fim-se
38.  Fim-para
39.  Para  $i \leftarrow 1$  até  $n_{inversao}$ 
40.      Para  $j \leftarrow 1$  até  $o$ 
41.          Selecionar pior  $W_j$ 
42.           $K_i \rightarrow W_j$ 
43.      Fim-Para
44.  Fim-Para
45.  Para  $i \leftarrow 1$  até  $n_{inversao}$  faça
46.       $sorteio \rightarrow$  Sorteio de um número de 1 até  $m$ 
47.       $A_{sorteio} \rightarrow K_i$ 
48.  Fim-para
49.  Para  $i \leftarrow 1$  até  $m$  faça
50.      Se  $f(X_i) < f(I^*)$  então
51.           $I^* \leftarrow X_i$ 
52.           $k \leftarrow 1$ 
53.      Fim-se
54.      Senão  $k \leftarrow k + 1$ 
55.  Fim-para
56.  Fim-enquanto
57.  Retorne  $I^*$ 

```

Fim AG

4 Resultados

Os programas desenvolvidos em C/C++ foram rodados para 15 problemas em um computador Core i5 3250M com 2GB de RAM. Os resultados da solução inicial com retorno estão na tabela abaixo.

Tabela 3 - Resultados metaheurísticas com método de solução inicial com retorno

Metaheurísticas			
Problemas	Busca Tabu	Algoritmo Genético	Melhor valor encontrado
M1a	6,337	6,310	6,310
M1b	6,505	6,505	6,505
M1c	6,410	6,398	6,398
M1d	7,172	6,408	6,408
M1e	6,670	7,175	6,670
M2a	7,611	7,669	7,611
M2b	7,842	8,030	7,842
M2c	7,787	7,787	7,787
M2d	7,523	7,544	7,523
M2e	7,523	7,573	7,523
M3a	10,415	10,350	10,350
M3b	10,397	10,277	10,277
M3c	10,190	10,286	10,190
M3d	10,208	10,263	10,208
M3e	10,063	9,792	9,792

Fonte: Autoria Própria

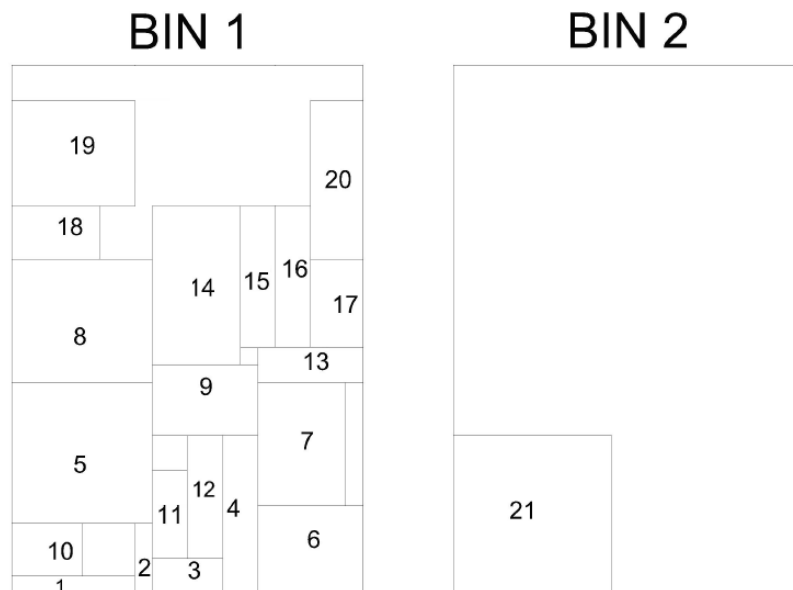
Como pode ser visto na Tabela 3, houve uma pequena variação dos resultados encontrados pelas duas metaheurísticas devido a aleatoriedade na escolha de alguns objetos do Algoritmo Genético. Foi observado que em ambos os casos este valor corresponde ao encontrado na solução inicial e se mantém como solução estrela até o final da execução das metaheurísticas. Pode-se concluir que na utilização do método de solução inicial com retorno, as metaheurísticas não conseguiram melhorar a solução encontrada pela heurística *Bottom-Left*.

No problema M1a resolvido pelas metaheurísticas, por exemplo, o método de geração da solução inicial com retorno pela heurística *Bottom-Left* utilizou 6 dos 16 *bins* disponíveis para alocação dos objetos. Como mostra a Tabela 1, os problemas da categoria M1 possuem 100 objetos. Estes são devidamente numerados e inseridos nos *bins*, dispostos em ordem decrescente de área. De acordo com a Tabela 2, temos que:

- a) *Bins* 1 e 2 – Possuem 600 unidades de área (20 de comprimento e 30 de altura);
- b) *Bins* 3, 4 e 5 – Possuem 400 unidades de área (10 de comprimento e 40 de altura);
- c) *Bin* 6 – Possui 400 unidades de área (20 de comprimento e 20 de altura).

A aleatoriedade para diferenciação dos indivíduos garante que 20% dos objetos sejam escolhidos por sorteio para serem encaixados primeiramente (genes iniciais do cromossomo de cada indivíduo) no método do Algoritmo Genético. A Figura 8 representa a visualização gráfica da solução inicial do Algoritmo Genético, mostrando o preenchimento dos *bins* 1 e 2 com os 20 objetos iniciais sorteados e o objeto subsequente não sorteado.

Figura 8 - Visualização Gráfica dos *bins* 1 e 2 do problema M1a com os 20 primeiros objetos inseridos por sorteio e o próximo objeto ordenado decrescente por área

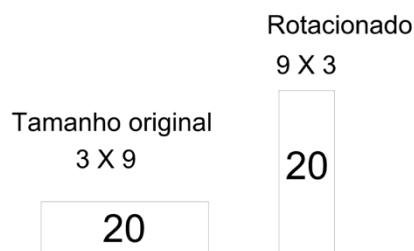


Fonte: Autoria Própria

Cada objeto busca o espaço disponível mais abaixo e mais a esquerda possível que consiga encaixá-lo. Para isto, cada unidade de área do *bin* é percorrida, da esquerda para a direita e de baixo para cima, até que um espaço vazio seja encontrado. Quando isto acontece, testa-se a ocupação de todas as unidades de área acima e a direita deste ponto que se equivalem a altura e ao comprimento do objeto, ou vice-versa. Isto garante que o objeto permita ser rotacionado em 90 graus.

O objeto 20, por exemplo, tem sua área definida no arquivo de entrada como 3x9 (altura x comprimento). Porém, teve que ser rotacionado para ser encaixado no *bin* 1.

Figura 9 - Exemplo de rotação de um objeto



Fonte: Autoria Própria

Os objetos 1, 2, 3 e 4, por exemplo, foram inseridos na base do *bin*. O objeto 5 não encontrou espaço suficiente no restante da base, ao lado do objeto 4. O ponto mais baixo e à esquerda do *bin* que o objeto 5 conseguiu encontrar foi acima do objeto 2 e à extremidade esquerda do *bin*, sobrando espaço suficiente para encaixe futuro do objeto 10. Esta lógica de alocação segue para todos os objetos do problema.

Após o encaixe dos 20 objetos iniciais, a ordem de inserção segue os objetos restantes em ordem decrescente de área. O objeto 21, com maior área dentre os restantes, percorre todo o *bin* 1 e não encontra nenhum espaço suficiente para armazená-lo. Desta forma, realiza seu encaixe no próximo *bin* disponível (*bin* 2), como pode ser visto na Figura 8.

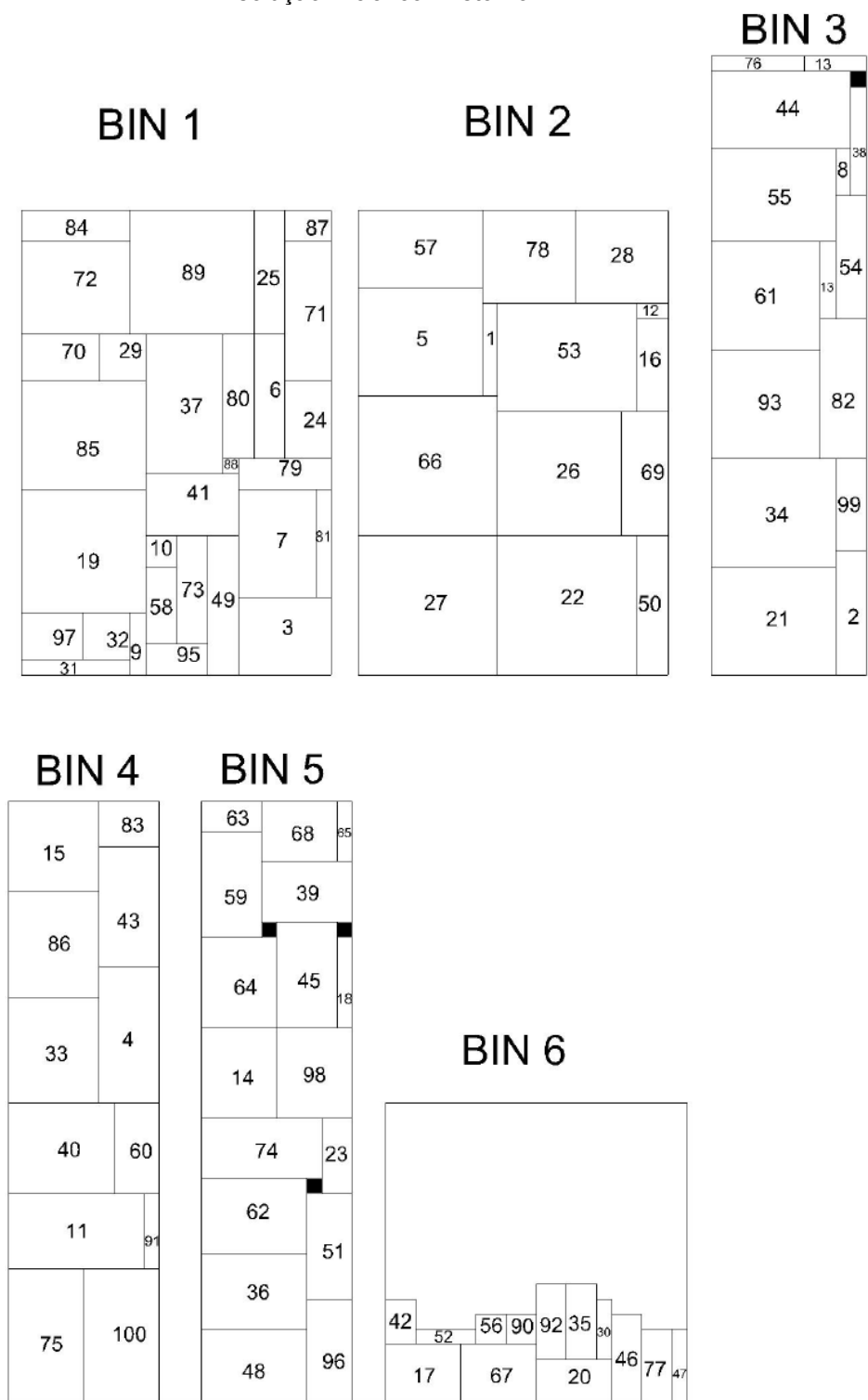
Os 6 *bins* preenchidos com os 100 objetos podem ser visualizados na Figura 10. Os 20 objetos sorteados representam, portanto, neste problema, os objetos de número: 31, 9, 95, 49, 19, 3, 7, 85, 41, 97, 58, 73, 79, 37, 80, 6, 24, 70, 72 e 71. O objeto de número 27 é o 21º objeto na ordem de inserção. Como o método de geração da solução inicial com retorno revisita todos os *bins* para a tentativa de encaixe dos próximos objetos, os *bins* 1, 2 e 4 têm sua área totalmente utilizada no decorrer da solução.

É possível notar que os *bins* de 1 a 5 possuem folga próximas ou iguais a zero. Assim, as metaheurísticas selecionam o *bin* 6 como *bin* de maior folga e tentam realizar a movimentação do objeto de menor área, mas não encontram área suficiente em nenhum dos *bins* anteriores.

Como o intuito do trabalho é mostrar o desempenho e a comparação entre as metaheurísticas, foi utilizado o método de geração da solução inicial sem retorno, garantindo uma solução pior que aquela gerada anteriormente para que as metaheurísticas tenham a chance de otimizá-la.

As seções 4.1 e 4.2 descrevem os testes realizados com o Busca Tabu e Algoritmo Genético, respectivamente. O resultado final e uma análise comparativa pode ser vista na seção 4.3.

Figura 10 - Visualização Gráfica da solução do problema M1a do Algoritmo Genético utilizando solução inicial com retorno



Fonte: Autoria Própria

4.1 Busca Tabu

Para a resolução dos 15 problemas propostos no trabalho, foram gerados resultados com 5 opções de tamanhos da lista tabu: 5%, 10%, 15%, 20% e 25% do número de objetos presentes no problema. Foi realizada essa análise para verificação se havia algum valor que se destacaria nas soluções obtidas. Após todas as instâncias estudadas com as respectivas porcentagens, pode-se perceber que não foi encontrada uma porcentagem que seria a mais adequada para todos os problemas, como pode ser visto na Tabela 4.

Tabela 4 - Soluções Busca Tabu e seus respectivos parâmetros

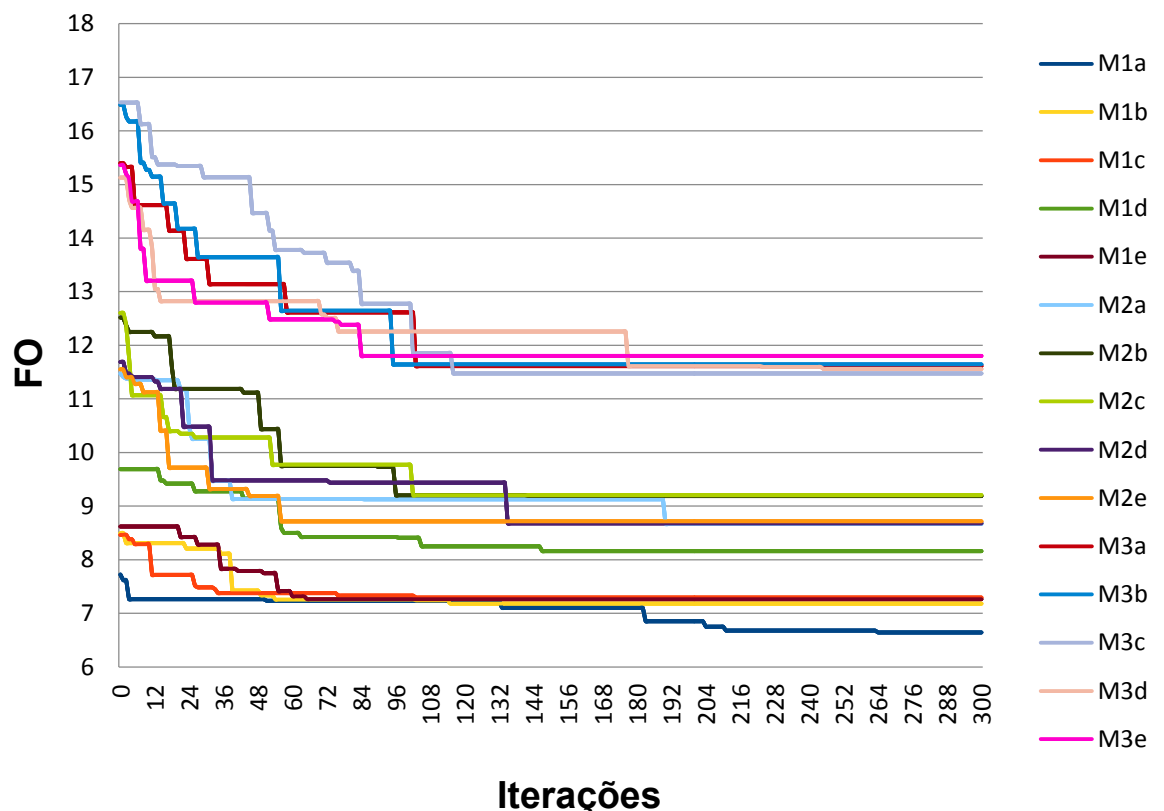
Problemas	Porcentagem Lista Tabu (%)					Melhor Porcentagem	Critério de parada (iterações)
	5	10	15	20	25		
M1a	7,12	6,64	6,68	6,69	7,09	10	265
M1b	7,44	7,31	7,22	7,25	7,18	25	116
M1c	7,47	7,29	7,33	7,33	7,33	10	104
M1d	8,27	8,25	8,17	9,69	9,69	15	148
M1e	7,71	7,37	7,26	7,37	8,67	15	66
M2a	9,67	9,55	9,67	9,23	8,67	25	90
M2b	11,51	9,22	9,75	9,19	9,58	20	143
M2c	10,19	10,19	9,20	9,28	9,77	15	103
M2d	11,40	8,74	8,68	8,68	9,31	15	101
M2e	9,69	8,72	8,72	8,72	8,72	25	57
M3a	13,08	12,61	11,61	12,61	12,61	15	104
M3b	12,66	13,30	11,64	12,66	12,66	15	96
M3c	13,57	12,57	11,48	13,57	18,69	15	117
M3d	11,56	11,82	11,82	12,18	11,82	5	246
M3e	13,13	12,12	11,80	11,80	11,80	20	85

Fonte: Autoria própria

A Tabela 4 ilustra essa heterogeneidade dos parâmetros com as soluções encontradas de alguns problemas, juntamente com o número de iterações sem melhora da solução estrela e a melhor porcentagem de lista tabu para o problema. Nos problemas onde houve valores iguais entre porcentagens de lista tabu, a melhor porcentagem foi escolhida entre a que encontrou aquele valor em um número menor de iterações, o que implica em um tempo computacional menor.

O critério de parada foi outro parâmetro que não foi possível determinar um valor base para todos os problemas. Cada problema teve um valor de iterações sem melhora como critério de parada encontrando o melhor valor possível nessas condições. O Gráfico 1 mostra a variação da FO por iteração para cada problema resolvido.

Gráfico 1 – FO por número de iteração para cada problema



Fonte: Autoria Própria

Através do Gráfico 1, nota-se o momento em que a solução não apresenta mais melhora para cada problema e este valor é definido como o critério de parada, mostrado na Tabela 4.

A metaheurística Busca Tabu implementada não possui variabilidade nos resultados, ou seja, todas as vezes que o código for rodado com os mesmos parâmetros, o mesmo resultado será obtido. O que diferencia os resultados é a modificação dos parâmetros. Por exemplo, no problema 1 (M1a), com 265 iterações como critério de parada foi encontrada uma solução de 6,64 *bins*. Com um número menor de iterações, o resultado é pior.

É perceptível também que a ordem de inserção dos objetos nos *bins* na solução inicial, influencia no resultado final encontrado, pois em todos os problemas estudados, o resultado final encontrado foi melhor quando os objetos foram inseridos em ordem decrescente de área, o que justifica a inserção dos objetos realizada, como pode ser visto na Tabela 5.

Tabela 5 - Comparação dos resultados em decorrência da ordem de inserção dos objetos

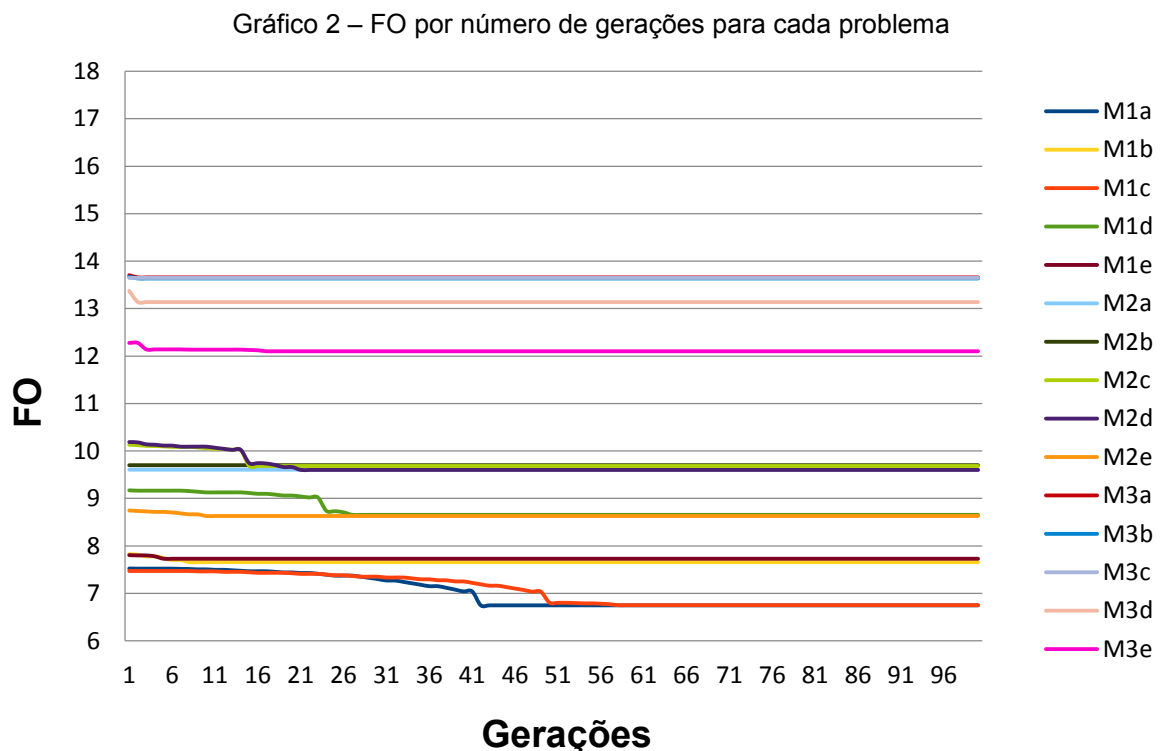
Problemas	Ordem de inserção	
	Ordem decrescente	Ordem do arquivo de entrada
M1a	6,64	7,36
M1b	7,18	7,35
M1c	7,29	7,43
M1d	8,17	8,40
M1e	7,26	8,20
M2a	9,13	10,51
M2b	9,19	11,36
M2c	9,28	10,59
M2d	8,68	9,66
M2e	8,72	9,47
M3a	11,61	15,21
M3b	11,64	15,40
M3c	11,48	14,56
M3d	11,56	15,47
M3e	11,80	12,54

Fonte: Autoria Própria

4.2 Algoritmo Genético

A metaheurística Algoritmo Genético, diferentemente da Busca Tabu, possui variabilidade nos resultados devido a aleatoriedade na criação dos cromossomos para diferenciação dos indivíduos. Assim, todas as vezes que o código for rodado com os mesmos parâmetros, o resultado obtido será diferente, porém com pequenas variações. Por outro lado, o algoritmo possui diferentes parâmetros que, combinados, alteram significativamente os resultados alcançados.

Os principais parâmetros para a execução do código do Algoritmo Genético são: número de indivíduos da população, número de gerações para definição do critério de parada e percentual de indivíduos mutados e cruzados. A fim de melhorar o desempenho da metaheurística, alguns testes foram realizados para definição dos parâmetros que possibilitaram melhores resultados de FO. O número de indivíduos da população foi limitado pela memória RAM do computador utilizado em um valor de 50 indivíduos.



Fonte: Autoria Própria

Os 15 problemas foram resolvidos com 100 gerações para análise da geração de estagnação da FO. Como pode ser visto no Gráfico 2, a partir de 60

gerações não houve alteração dos valores em nenhum dos problemas. Portanto, 60 gerações foi estabelecido como parâmetro de critério de parada para a metaheurística Algoritmo Genético.

Considerando então 60 gerações de populações com 50 indivíduos, os 15 problemas foram resolvidos três vezes cada um para todos os possíveis pares de percentuais de ocorrência de mutação e cruzamento entre os indivíduos.

A Tabela 6 mostra os resultados encontrados para cada resolução do problema M1a.

Tabela 6 - Problema M1a para cada percentual de mutação e crossover

Porcentagem (%)		Rodada			
Mutação	Crossover	1	2	3	Média
10	90	7.2325	7.0600	7.1700	7.1542
20	80	6.7317	6.6400	7.6167	6.9961
30	70	6.8267	6.7317	6.7050	6.7544
40	60	6.7900	6.7800	6.6183	6.7294
50	50	6.8267	7.4575	6.7250	7.0031
60	40	6.7483	7.3225	6.7467	6.9392
70	30	6.7075	6.7767	6.7467	6.7436
80	20	7.4400	7.2325	7.3375	7.3367
90	10	7.5483	7.4375	7.3925	7.4594

Fonte: Autoria Própria

Os possíveis pares de percentuais de mutação e crossover estão listados na Tabela 6 e as três rodadas representam os resultados de FO encontrados para cada vez que o problema foi resolvido. A média para cada possibilidade foi calculada e em negrito pode-se visualizar a menor média encontrada. Para este problema, por exemplo, foram definidos os percentuais de 40% e 60% de

ocorrência de mutação e crossover, respectivamente. Os resultados encontrados para as três rodadas de cada problema podem ser vistos no Apêndice A. A Tabela 7 mostra um resumo dos percentuais de mutação e crossover definidos para cada um dos 15 problemas.

Tabela 7 - Percentuais de mutação e crossover para cada problema

Porcentagem (%)		
Mutação	Crossover	Problemas
10	90	-
20	80	6
30	70	-
40	60	1,3,4,10,11,15
50	50	2,9
60	40	5,7,12,13,14
70	30	-
80	20	8
90	10	-

Fonte: Autoria própria

Nota-se que a combinação de 40% de mutação e 60% de crossover obteve melhores resultados para 6 dos 15 problemas resolvidos. Em contrapartida, 5 dos problemas obtiveram melhores resultados considerando 60% e 40% de mutação e crossover, respectivamente. É perceptível então que o eixo central dos pares de mutação e crossover, ou seja, a região onde a diferença entre eles é menor, é geralmente onde se encontra as melhores soluções por aumentar a variabilidade genética dos indivíduos gerados.

Com o número de indivíduos da população, o número de gerações e os percentuais de mutação e reprodução definidos, os resultados obtidos são utilizados para a comparação entre as duas metaheurísticas.

4.3 Busca Tabu x Algoritmo Genético

Com os parâmetros de cada metodologia definidos e os testes realizados, os resultados finais encontrados estão dispostos na Tabela 8.

Tabela 8 - Resultados das metaheurísticas com método de solução inicial sem retorno

Problemas	Busca Tabu		Algoritmo Genético		Limitante Inferior
	FO	Tempo (s)	FO	Tempo (s)	
M1a	6,642	0,089	6,618	56,282	6
M1b	7,182	0,052	6,788	71,813	6
M1c	7,292	0,020	6,710	66,157	6
M1d	8,170	0,059	8,600	68,129	7
M1e	7,263	0,021	7,542	60,657	6
M2a	9,127	0,345	8,637	581,712	7
M2b	9,189	0,537	9,536	538,507	7
M2c	9,285	0,307	9,391	561,556	7
M2d	8,677	0,348	9,592	516,999	7
M2e	8,716	0,300	8,513	499,568	7
M3a	11,612	0,567	12,606	1007,060	9
M3b	11,644	0,857	12,640	912,902	9
M3c	11,477	1,064	12,737	868,167	9
M3d	11,560	1,903	12,292	899,462	9
M3e	11,801	0,395	11,336	889,824	9

Fonte: Autoria Própria

O limitante inferior presente na tabela é encontrado através da soma de todas as áreas dos objetos e então soma-se a área dos *bins*, começando pelos de maior área, encontrando quantos *bins* seriam necessários para armazenar a área total dos objetos.

Os valores em **negrito** representam os melhores valores de FO encontrados entre a Busca Tabu e o Algoritmo Genético. Nota-se que nos problemas M1a, M1d, M1e, M2b, M2c, M2e, e M3e, a solução final encontrada para o número de *bins* utilizados em ambas metaheurísticas foi igual, diferenciando apenas pelo fator de utilização do *bin* mais vazio (parte decimal do resultado). Nestes problemas então, considera-se que a Busca Tabu foi superior, encontrando os mesmos resultados do Algoritmo Genético em um tempo computacional consideravelmente menor. Nos outros problemas, a melhor metaheurística foi a que encontrou o menor número de *bins*, independente do tempo necessário, pois mesmo o Algoritmo Genético levando mais tempo que a Busca Tabu para terminar sua execução, esse período é viável pois não ultrapassa 20 minutos (o problema M3a foi o problema em que foi gasto o maior tempo computacional, com aproximadamente 17 minutos).

A visualização gráfica das soluções do problema M1a geradas pelo Algoritmo Genético e Busca Tabu utilizando as soluções iniciais com retorno e sem retorno podem ser visualizadas no apêndice B do trabalho. Através dessas visualizações podemos verificar a aleatoriedade dos objetos iniciais na solução inicial com retorno do Algoritmo Genético, com os objetos seguintes inseridos sequencialmente em ordem decrescente de área e a inserção decrescente de área dos objetos desde o primeiro até o último pela solução inicial com retorno da Busca Tabu. Nas soluções sem retorno pode ser analisada a atuação das metaheurísticas realizando modificações nas soluções buscando uma solução viável em tempo computacional reduzido.

No caso do problema M1a nota-se que o resultado foi satisfatório pois encontrou o mesmo número de *bins* da solução inicial com retorno produzida pela heurística *Bottom-Left*. Já nos outros problemas estudados esse fato não se repetiu, ocasionando um número maior de *bins* nas soluções sem retorno em comparação com as com retorno.

5 Conclusão

O tamanho da lista tabu de 15% do número de objetos mostrou ser o mais adequado para 46,67% dos problemas resolvidos com a metaheurística Busca Tabu. Tamanho de lista abaixo de 15% não se mostrou tão eficiente, se destacando em apenas 20% dos casos. Isso acontece porque com listas tabu maiores o número de soluções diferentes percorridas é maior aumentando a possibilidade de encontrar melhores soluções. Listas maiores que 25% não foram viáveis pois os objetos menores se tornam proibidos forçando a movimentação dos objetos maiores que é bem mais dificultosa.

A prévia ordenação dos objetos em ordem decrescente de área para inserção nos *bins* na busca da solução inicial gerou melhores resultados em 100% dos casos em comparação àqueles que inseriam os objetos de acordo com a ordem de disposição nos arquivos de entrada.

Os percentuais de mutação e crossover do Algoritmo Genético foram testados e observou-se que as combinações que obtiveram os melhores resultados foram aquelas utilizando 40% e 60% de mutação, e cada uma delas obtiveram os melhores resultados em 40% e 33,33% dos problemas, respectivamente, pelo fato de gerar maior variabilidade genética em comparação com porcentagens pequenas de mutação ou crossover.

O Algoritmo Genético mostrou estagnação dos resultados desde a 3ª até 60ª geração para os problemas resolvidos. Os problemas do grupo M3 que possuem objetos de maior área e, conseqüentemente, maior dificuldade de resolução, tenderam a estagnação precoce, pois na mutação dificilmente um objeto de grandes dimensões conseguia ser movido, encontrando uma dependência maior dos resultados do crossover.

A Busca Tabu obteve um desempenho superior ao Algoritmo Genético, com melhores resultados em 12 dos 15 problemas. O Algoritmo Genético apresentou melhores resultados em 2 problemas da categoria M1 e em 1 problema da categoria M2, sendo superado pela Busca Tabu em todos os problemas da categoria M3, ou seja, os problemas com maior número de objetos e com as maiores dimensões dos objetos e dos *bins*.

O Algoritmo Genético leva vantagem em relação a Busca Tabu no que diz respeito a velocidade de afunilamento dos resultados, como foram vistos nos Gráficos 1 e 2, pelo fato de se basear em um conjunto de soluções. Porém, apesar da mutação se assemelhar à movimentação do método Busca Tabu, este não realiza a proibição dos movimentos recentes realizados, o que gera uma rápida estagnação dos resultados devido a repetições na tentativa de movimentação do mesmo objeto. Além disso, a criação de várias soluções por geração fez com que o código do Algoritmo Genético utilizasse um tempo computacional maior atingindo uma diferença de até 1800 vezes superior ao utilizado pela Busca Tabu.

O objetivo do trabalho foi mostrar a diferença no desempenho das metaheurísticas Busca Tabu e Algoritmo Genético na resolução de problemas bidimensionais de *bin packing*. Na comparação entre elas a metaheurística Busca Tabu se sobressaiu em relação ao Algoritmo Genético, principalmente em problemas de difícil resolução. Contudo, na utilização da heurística de encaixe *Bottom-Left* para geração da solução inicial com retorno em problemas de até 150 objetos, pode-se perceber a eficiência deste método em relação as metaheurísticas estudadas para a resolução de problemas de *bin packing* bidimensionais. A ordenação dos bins em ordem decrescente de área nos arquivos de entrada pode ter sido o fator que acarretou essa vantagem da heurística percebida nos resultados por dificultar a modificação de soluções diminuindo o número de *bins* da solução inicial. Em trabalhos futuros, testes podem ser realizados para categorias com maior número e dimensão de objetos e além disso, respeitar a ordem dos *bins* como estão dispostos no problema.

Referências Bibliográficas

- CONSTANTINO, AA. & JUNIOR, AMG. 2002. **Um algoritmo genético híbrido para o problema de corte industrial bidimensional**. Acta Scientiarum v. 24, n. 6, p. 1727-1731, Maringá, PR, Brasil.
- HIGGINS, A.J. 2001. **A dynamic tabu search for large-scale generalized assignment problems**. Computers and Operations Research, v. 28, p. 1039-1048.
- LUCAS, DC. 2002. **Algoritmos Genéticos: uma Introdução**. Apostila de Ferramentas de Inteligência Artificial, RS, Brasil.
- MANSANO, EM. 2008. **Algoritmos Busca Tabu paralelos aplicados ao planejamento da Expansão da Transmissão de Energia Elétrica**. Dissertação (Mestrado em Engenharia Elétrica-Automação), UNESP-FEIS, Ilha Solteira, SP, Brasil.
- MARTÍNEZ, DA. 2014. **Estudo dos problemas de corte e empacotamento**. Dissertação (Doutorado em Engenharia Elétrica - Automação), UNESP-FEIS, Ilha Solteira, SP, Brasil.
- OR-Library**. Disponível em: <<http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/>>. Acesso em: 1 mai. 2017.
- PEREIRA, JMRP. 2013. **Uma heurística de melhor encaixe para um problema de empacotamento a duas dimensões**. Dissertação de Mestrado em Engenharia de Sistemas, Universidade do Minho, Braga, Portugal.
- REIS, JVA. & CUNHA, CB. 2009. **Uma heurística baseada em busca em vizinhança variável para o problema de agrupamento de entregas em veículos de uma frota heterogênea**. Congresso de Pesquisa e Ensino em Transportes, 1-12.
- SRIDHAR, R.; CHANDRASEKARAN, M.; SRIRAMYA, P.; PAGE, T. 2017. **Optimization of heterogeneous Bin packing using adaptive genetic algorithm**. International Conference on Emerging Trends in Engineering Research, TN, India.

SUBRAMANIAN, A.; MEDEIROS, JMF.; FORMIGA, LA.; SOUZA, MJF. 2011. **Aplicação da metaheurística Busca Tabu ao problema de alocação de aulas a salas em uma instituição universitária.** Revista Produção Online, v.11, n.1, p. 54-75.

SUCUPIRA, IR. 2004. **Métodos heurísticos genéricos: Meta-heurísticas e hiper-heurísticas.** Universidade de São Paulo, SP, Brasil.

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

APÊNDICE A – Soluções encontradas para cada problema testando pares de porcentagens no Algoritmo Genético

Tabela 9 - Problema M1a para cada percentual de mutação e crossover

Porcentagem (%)		Rodada			
Mutação	Crossover	1	2	3	Média
10	90	7.2325	7.0600	7.1700	7.1542
20	80	6.7317	6.6400	7.6167	6.9961
30	70	6.8267	6.7317	6.7050	6.7544
40	60	6.7900	6.7800	6.6183	6.7294
50	50	6.8267	7.4575	6.7250	7.0031
60	40	6.7483	7.3225	6.7467	6.9392
70	30	6.7075	6.7767	6.7467	6.7436
80	20	7.4400	7.2325	7.3375	7.3367
90	10	7.5483	7.4375	7.3925	7.4594

Fonte: Autoria Própria

Tabela 10 - Problema M1b para cada percentual de mutação e crossover

Porcentagem (%)		Rodada			
Mutação	Crossover	1	2	3	Média
10	90	7.6700	7.6500	7.6783	7.6661
20	80	7.7300	7.6333	7.6600	7.6744
30	70	7.6367	7.6033	6.7925	7.3442
40	60	7.6600	7.6783	7.5000	7.6128
50	50	6.7883	7.6183	7.5900	7.3322
60	40	7.6633	7.7317	7.6517	7.6822
70	30	7.7033	7.6517	7.6450	7.6667
80	20	7.7467	7.6917	7.5200	7.6528
90	10	7.6450	7.6450	7.7817	7.6906

Fonte: Autoria Própria

Tabela 11 - Problema M1c para cada percentual de mutação e crossover

Porcentagem (%)		Rodada			
Mutação	Crossover	1	2	3	Média
10	90	7.5325	7.4275	7.3225	7.4275
20	80	7.5700	7.5700	7.5700	7.5700
30	70	7.5325	7.5550	7.6200	7.5692
40	60	6.7383	6.7100	6.8350	6.7611
50	50	6.8167	7.4350	6.7783	7.0100
60	40	7.5550	7.4425	7.5325	7.5100
70	30	7.6633	6.7375	7.5475	7.3161
80	20	7.5400	6.7433	7.6200	7.3011
90	10	7.4575	7.3825	7.4300	7.4233

Fonte: Autoria Própria

Tabela 12 - Problema M1d para cada percentual de mutação e crossover

Porcentagem (%)		Rodada			
Mutação	Crossover	1	2	3	Média
10	90	8.6475	8.7467	8.6667	8.6870
20	80	8.7200	8.7200	8.5617	8.6672
30	70	8.7200	8.7200	8.7067	8.7156
40	60	8.7067	8.6000	8.6800	8.6622
50	50	8.7133	8.6150	8.6900	8.6728
60	40	8.5860	8.6800	8.7600	8.6753
70	30	8.6325	8.6600	8.7067	8.6664
80	20	8.7200	8.6800	8.7067	8.7022
90	10	9.1333	9.1967	8.6667	8.9989

Fonte: Autoria Própria

Tabela 13 - Problema M1e para cada percentual de mutação e crossover

Porcentagem (%)		Rodada			
Mutação	Crossover	1	2	3	Média
10	90	7.7233	7.5600	7.7367	7.6733
20	80	7.6867	7.6633	7.6300	7.6600
30	70	7.7250	7.6675	7.6950	7.6958
40	60	7.7100	7.6917	7.7100	7.7039
50	50	7.7230	7.7060	7.6900	7.7063
60	40	7.6730	7.5883	7.5417	7.6010
70	30	7.7080	7.6700	7.7383	7.7054
80	20	7.7250	7.7500	7.6500	7.7083
90	10	7.7920	8.1033	7.6900	7.8618

Fonte: Autoria Própria

Tabela 14 - Problema M2a para cada percentual de mutação e crossover

Porcentagem (%)		Rodada			
Mutação	Crossover	1	2	3	Média
10	90	9.6081	9.5781	9.4880	9.5581
20	80	8.6372	8.7444	9.0338	8.8051
30	70	9.6372	9.7233	9.7107	9.6904
40	60	9.6297	9.5053	9.6297	9.5882
50	50	9.6139	9.6139	9.6056	9.6111
60	40	9.6297	9.4594	9.4933	9.5275
70	30	9.4590	9.6214	9.6372	9.5725
80	20	9.6372	9.5889	9.5178	9.5813
90	10	9.6472	9.6472	9.6497	9.6481

Fonte: Autoria Própria

Tabela 15 - Problema M2b para cada percentual de mutação e crossover

Porcentagem (%)		Rodada			
Mutação	Crossover	1	2	3	Média
10	90	9.7344	9.7328	9.6691	9.7121
20	80	10.3733	10.3375	10.3650	10.3586
30	70	9.6372	9.6606	9.6606	9.6528
40	60	9.6174	9.7328	9.6447	9.6650
50	50	9.7083	9.6139	9.7189	9.6804
60	40	9.6607	9.5361	9.7033	9.6334
70	30	9.7344	9.7344	9.7344	9.7344
80	20	9.5889	10.5893	10.3733	10.1838
90	10	10.5281	10.1025	10.4894	10.3734

Fonte: Autoria Própria

Tabela 16 - Problema M2c para cada percentual de mutação e crossover

Porcentagem (%)		Rodada			
Mutação	Crossover	1	2	3	Média
10	90	9.5859	9.6839	9.5756	9.6151
20	80	9.6822	9.6572	9.7311	9.6902
30	70	9.6289	9.7000	9.6600	9.6630
40	60	9.7311	9.7311	9.5806	9.6809
50	50	9.7311	9.6314	9.6206	9.6610
60	40	9.6106	9.5513	9.6820	9.6146
70	30	9.7070	9.7024	9.6474	9.6856
80	20	9.6633	9.6911	9.3908	9.5818
90	10	9.5606	10.0225	10.1638	9.9156

Fonte: Autoria Própria

Tabela 17 - Problema M2d para cada percentual de mutação e crossover

Porcentagem (%)		Rodada			
Mutação	Crossover	1	2	3	Média
10	90	10.0450	10.0525	10.0600	10.0525
20	80	9.6750	9.6325	9.5917	9.6331
30	70	9.6625	9.6750	9.6650	9.6675
40	60	9.7200	9.6417	9.7200	9.6939
50	50	9.6525	9.6450	9.5917	9.6297
60	40	9.6200	9.6000	9.7200	9.6467
70	30	9.6450	9.6900	9.7200	9.6850
80	20	9.6025	9.6400	9.6600	9.6342
90	10	10.2025	10.1200	10.1500	10.1575

Fonte: Autoria Própria

Tabela 18 - Problema M2e para cada percentual de mutação e crossover

Porcentagem (%)		Rodada			
Mutação	Crossover	1	2	3	Média
10	90	9.1090	8.7175	8.7083	8.8449
20	80	8.7920	8.7574	8.6700	8.7398
30	70	8.7900	8.7531	8.6908	8.7447
40	60	8.5860	8.5126	8.5126	8.5371
50	50	8.7810	8.5206	8.6235	8.6417
60	40	8.7810	8.6289	8.7898	8.7332
70	30	8.7830	8.7058	8.7025	8.7304
80	20	8.6580	8.5563	8.5570	8.5904
90	10	8.7430	9.1050	9.0300	8.9593

Fonte: Autoria Própria

Tabela 19 - Problema M3a para cada percentual de mutação e crossover

Porcentagem (%)		Rodada			
Mutação	Crossover	1	2	3	Média
10	90	13.7332	13.3867	13.7037	13.6078
20	80	13.7020	13.3867	13.1289	13.4058
30	70	13.3967	13.6213	13.7038	13.5739
40	60	12.7003	12.6061	13.4783	12.9283
50	50	13.5050	13.3867	13.3866	13.4261
60	40	13.2578	13.6925	13.3867	13.4456
70	30	13.6590	13.5343	13.6633	13.6189
80	20	13.5028	13.3867	13.7332	13.5409
90	10	13.3611	13.5806	13.6967	13.5461

Fonte: Autoria Própria

Tabela 20 - Problema M3b para cada percentual de mutação e crossover

Porcentagem (%)		Rodada			
Mutação	Crossover	1	2	3	Média
10	90	13.6493	13.6542	13.5133	13.6056
20	80	12.7485	13.5890	12.7105	13.0160
30	70	12.7178	13.5133	13.3867	13.2059
40	60	12.6144	13.3233	13.6715	13.2031
50	50	12.7308	13.1289	13.2790	13.0462
60	40	13.6603	12.6938	12.6400	12.9981
70	30	13.6440	13.6627	12.5133	13.2733
80	20	13.6340	12.7568	13.3856	13.2588
90	10	13.3395	13.5144	13.2723	13.3754

Fonte: Autoria Própria

Tabela 21 - Problema M3c para cada percentual de mutação e crossover

Porcentagem (%)		Rodada			
Mutação	Crossover	1	2	3	Média
10	90	13.5833	13.6595	13.5708	13.6046
20	80	13.6315	13.6595	13.6720	13.6543
30	70	13.7372	13.5833	13.4950	13.6052
40	60	13.5568	13.2639	13.6175	13.4794
50	50	13.6875	13.6735	13.5128	13.6246
60	40	13.5911	13.6565	12.7372	13.3283
70	30	13.6872	13.6417	13.6433	13.6574
80	20	13.6450	13.6417	13.6417	13.6428
90	10	13.5553	13.6595	13.5833	13.5994

Fonte: Autoria Própria

Tabela 22 - Problema M3d para cada percentual de mutação e crossover

Porcentagem (%)		Rodada			
Mutação	Crossover	1	2	3	Média
10	90	12.7545	12.6798	12.5045	12.6463
20	80	12.7592	12.6080	12.7592	12.7088
30	70	12.6980	12.7088	12.7067	12.7045
40	60	12.7067	12.5028	12.6798	12.6298
50	50	12.5425	12.7840	12.4988	12.6084
60	40	12.2917	12.3330	12.4350	12.3532
70	30	12.7357	12.7357	12.6798	12.7171
80	20	13.1360	12.7838	13.1140	13.0113
90	10	13.1600	13.0160	13.0820	13.0860

Fonte: Autoria Própria

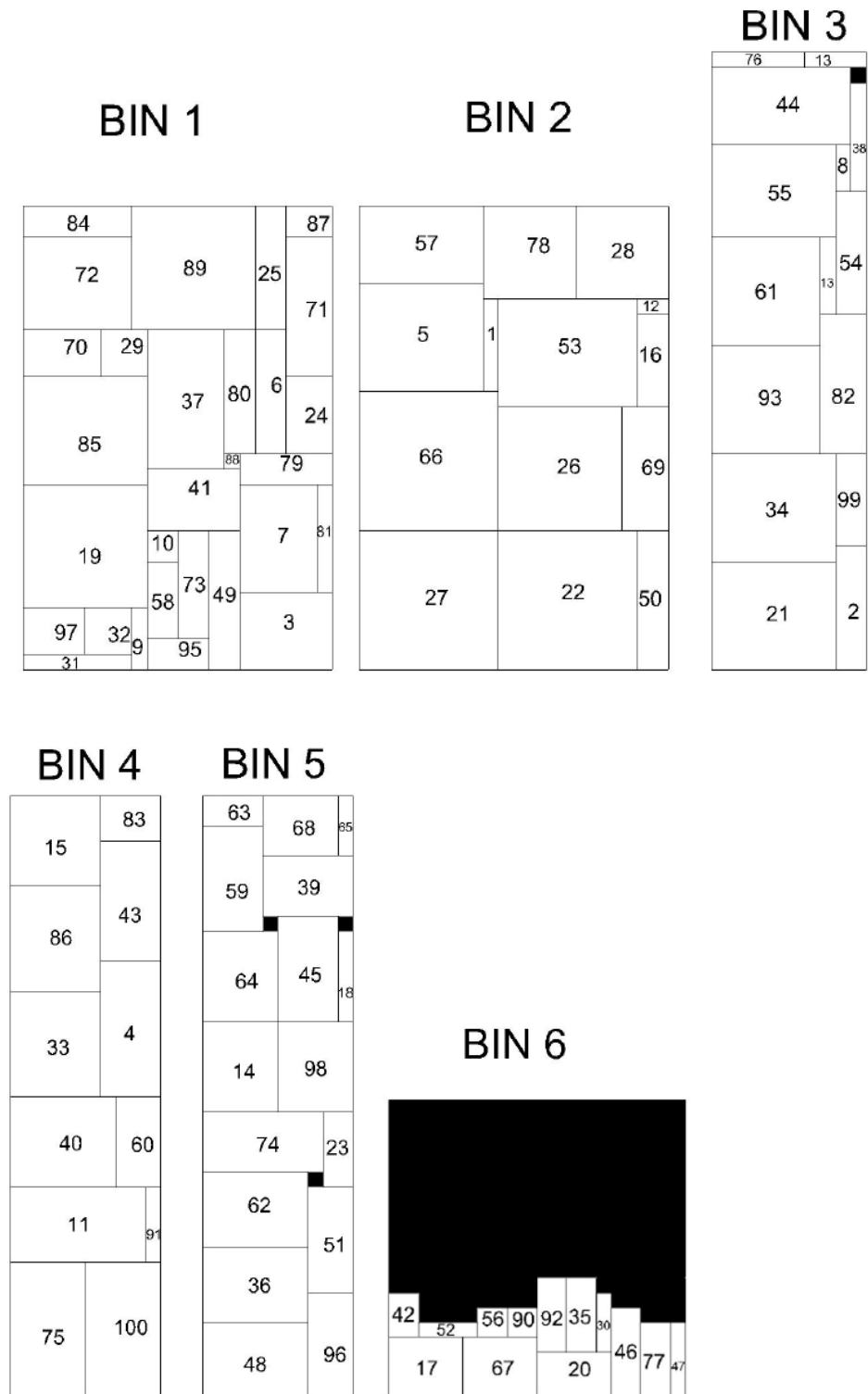
Tabela 23 - Problema M3e para cada percentual de mutação e crossover

Porcentagem (%)		Rodada			
Mutação	Crossover	1	2	3	Média
10	90	12.0570	11.8300	12.2163	12.0344
20	80	11.8300	11.6950	11.8200	11.7817
30	70	11.6758	11.5481	11.6067	11.6102
40	60	11.5630	11.5669	11.3360	11.4880
50	50	11.6560	11.7252	11.6903	11.6905
60	40	11.4560	11.7089	11.7770	11.6473
70	30	11.6570	11.8400	11.6731	11.7234
80	20	11.5672	12.0130	11.4778	11.6860
90	10	12.1360	12.1393	12.3475	12.2076

Fonte: Autoria Própria

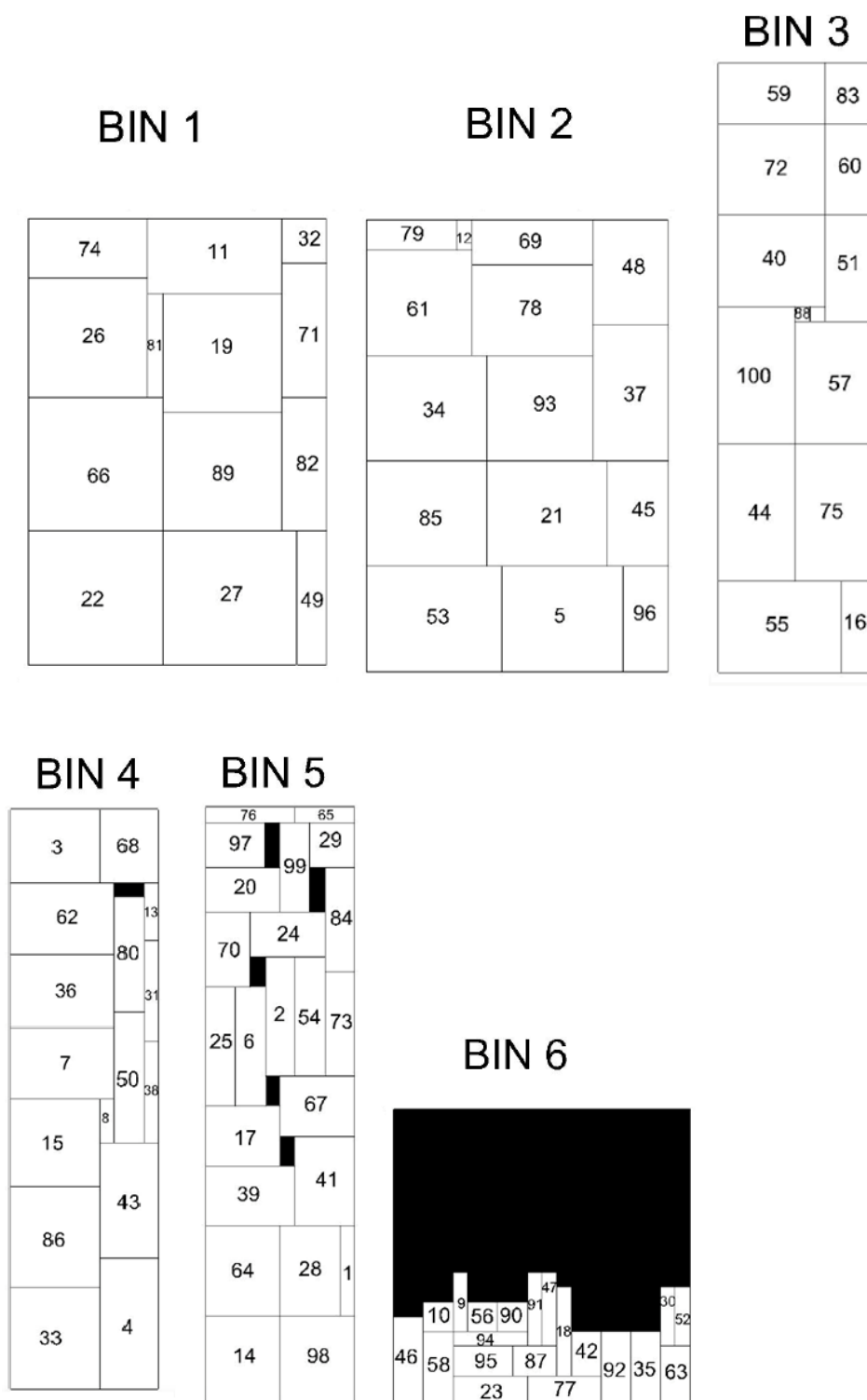
APÊNDICE B – Visualizações gráficas do problema M1a para a Busca Tabu e o Algoritmo Genético utilizando as soluções iniciais com retorno e sem retorno

Figura 11 - Visualização gráfica Algoritmo Genético utilizando solução inicial com retorno



Fonte: Autoria Própria

Figura 12 - Visualização gráfica Busca Tabu utilizando solução inicial com retorno



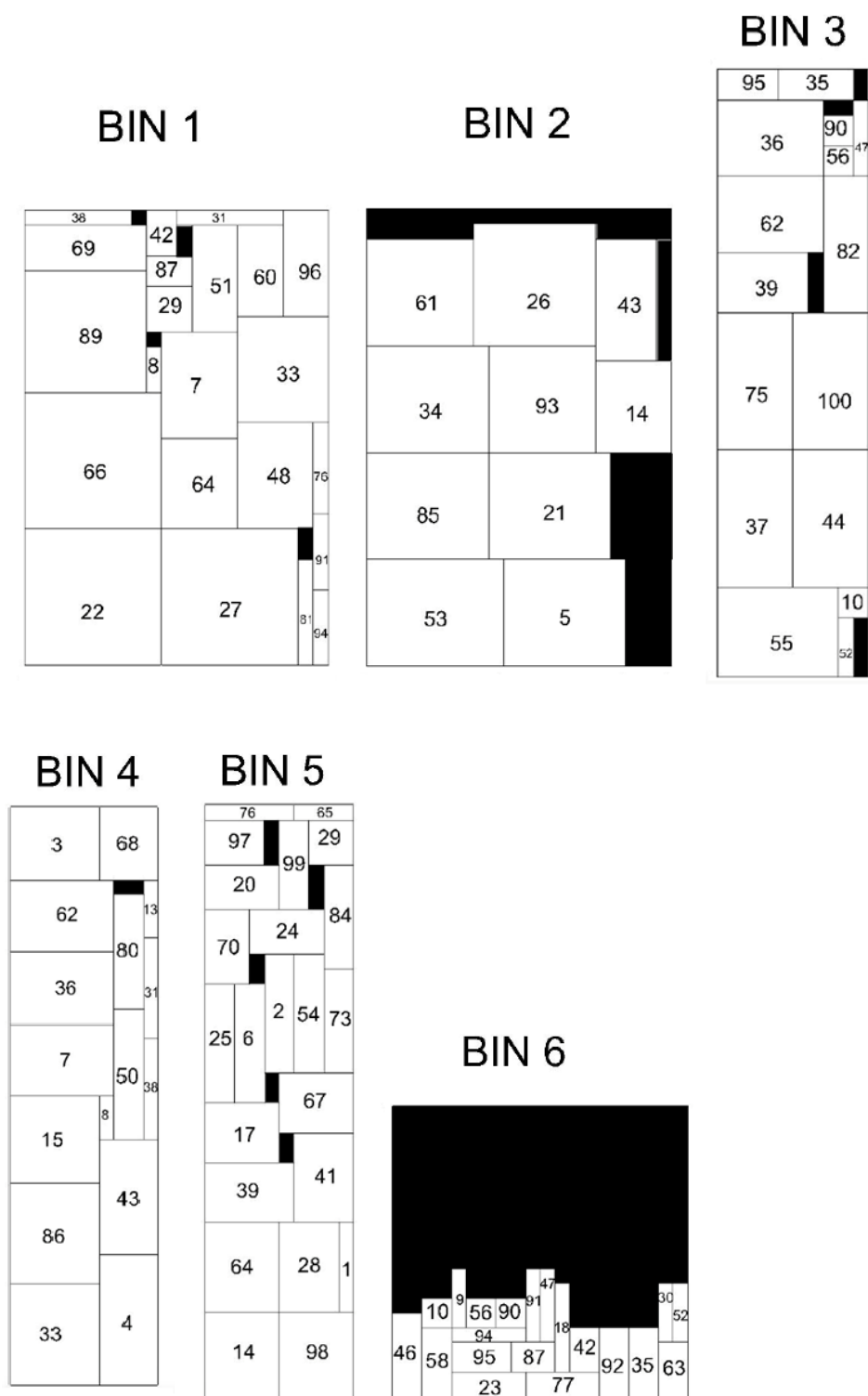
Fonte: Autoria Própria

Figura 13 - Visualização gráfica Algoritmo Genético utilizando solução inicial sem retorno



Fonte: Autoria Própria

Figura 14 - Visualização gráfica Busca Tabu utilizando solução inicial sem retorno



Fonte: Autoria Própria