Algoritmo Genético Aplicado ao Problema de Empacotamento

Antonia Raiane Santos Araujo Cruz[1], **Antonio Alex de Souza**[1], **Wanessa de Caldas Teotônio**[1]

[1]Mestrado Acadêmico em Ciência da Computação - Universidade Estadual do Ceará (UECE)

raianeti@gmail.com, alex.souza@aluno.uece.br, wanessacaldast@gmail.com

Abstract. *Bin Packing is a class of combinatorial optimization problems, which relate a finite set of elements with one or more constraints. In the case of the problem presented in this article, the objective is to allocate a certain amount of items in the smallest number of boxes, considering the weight of the items and the capacity supported by the bin. Since this problem is of the NP-Difficult type, that is, the execution time to find an optimal solution is of exponential growth, some methods have been proposed to solve it in a shorter time and presenting a solution that can be approximated to the optimal . These methods are called metaheuristics. Among the existing meta-heuristics, this article presents the use of the meta-heuristic Genetic Algorithm to solve the Bin Packing problem. Based on the experiments performed, the performance of the genetic algorithm was evaluated in different instances of the problem. Finally, it was found that the results presented were satisfactory.*

Resumo. *Bin Packing é uma classe de problemas de otimização combinatória, que relaciona um conjunto finito de elementos com uma ou mais restrições. No caso do problema apresentado neste artigo, o objetivo é alocar uma quantidade determinada de itens no menor número de caixas, considerando o peso dos itens e a capacidade suportada pela caixa. Visto que este problema é do tipo NP-Difícil, ou seja, o tempo de execução para encontrar uma solução ótima é de crescimento exponencial, alguns métodos têm sido propostos para resolvê-lo em tempo menor e apresentando uma solução que pode estar aproximada da ótima. Esses métodos são chamados de métodos heurísticos. Dentre os métodos heurísticos existentes, este artigo apresenta o uso da meta-heurística Algoritmo Genético para solucionar o problema Bin Packing. Com base nos experimentos realizados, avaliou-se a performance do algoritmo genético em diferentes instâncias do problema. Por fim, constatou-se que os resultados apresentados foram satisfatórios quando comparados aos resultados ótimos conhecidos.*

1. Introdução

O problema *Bin Packing*, também conhecido como Problema do Empacotamento, é uma classe de problemas bastante comum que, como muitos de natureza combinatória, esconde por trás de sua aparente simplicidade (facilmente formulados e compreendidos), a sua real complexidade. Esse problema apresenta um conjunto finito de elementos, que possuem um determinado volume, e precisam ser alocados em um número mínimo de recipientes, de forma que a soma do volume dos elementos não ultrapasse a capacidade suportada pelo recipiente.

Este problema é do tipo NP-Difícil, ou seja, o tempo de execução para encontrar uma solução ótima é de crescimento exponencial. Conforme Garey e Johnson (1979) é

pouco provável que exista um algoritmo de otimização em tempo polinomial para problemas do tipo NP-Difícil. Portanto, alguns métodos têm sido propostos para resolvê-los em tempo menor e apresentando uma solução que pode estar aproximada da solução ótima. Esses métodos são chamados de métodos heurísticos. Dentre os métodos heurísticos existentes, a meta-heurística Algoritmo Genético (AG) é uma proposta para solucionar o problema *Bin Packing*. Recomenda-se a meta-heurística AG pelos bons resultados, tanto para problemas de corte e empacotamento quanto para outros problemas da classe NP-difícil [George and Robinson 1980]; [Morabito and Arenalest 1994].

Heurísticas são técnicas desenvolvidas para solucionar problemas específicos, enquanto as meta-heurísticas são de uso geral e podem ser utilizadas em diferentes classes de problemas, comumente encontrados em áreas de indústrias, tais como: metalúrgica, papel, vidro, plástica e têxtil [Cogo and Furtado 2001].

O presente trabalho tem como objetivo apresentar uma proposta de solução para o Problema de Empacotamento utilizando a meta-heurística AG, com a finalidade de retornar uma solução satisfatória para o problema proposto.

Desse modo, este artigo está estruturado da seguinte forma: a seção 2 apresenta a fundamentação teórica e contextualização do problema; a seção 3 descreve a abordagem proposta, as instâncias do problema, configurações, entre outras; a seção 4 apresenta o desenvolvimento e análise dos experimentos e resultados obtidos; e, por fim, a seção 5, discorre sobre as considerações finais e sugestões de trabalhos futuros provenientes desta pesquisa.

2. Fundamentação Teórica

2.1. Otimização Combinatória

A área da Otimização Combinatória abrange estudos matemáticos para encontrar um arranjo, agrupamento ou seleção ótima de objetos discretos. Relaciona-se diretamente com a tomada de decisões em problemas de diversas áreas do conhecimento, por exemplo, problemas de planejamento e programação da produção, problemas de corte e empacotamento, entre outros. Ao considerar alguns desses problemas, surgem, frequentemente, vários critérios de desempenho (funções objetivos), em geral conflitantes entre si [Brown 1988].

Segundo Arroyo et al. (2002), os problemas de otimização multiobjetivo consiste em encontrar um vetor de variáveis de decisão (solução) satisfazendo as restrições impostas e otimizando uma função vetorial. As técnicas que envolvem este modelo de otimização buscam encontrar uma solução que satisfaça algumas restrições e otimize uma função objetivo, que é responsável por avaliar a qualidade destas soluções.

As soluções baseadas em Algoritmo Genético exploram uma população de resultados a cada iteração, de forma que as estratégias de buscas utlizadas por este mecanismo permitem explorar várias regiões do espaço de soluções de cada vez. Portanto, ao longo das iterações não se constrói uma única trajetória de busca, pois novas saídas sempre são obtidas através das combinações das soluções anteriores.

2.2. Problema de Empacotamento (*Bin Packing Problem*)

Dentre os problemas clássicos da Otimização Combinatória está o Problema de Empacotamento, *Bin Packing Problem*, extensivamente estudado por ser um problema de difícil

solução, onde a possibilidade de existir algoritmos exatos que o resolva, em tempo de execução razoável, é muito pequena [Faria 2016]. Portanto, é necessário a aplicação de métodos heurísticos e algoritmos aproximativos que demandam por soluções muito próximas do ótimo e com tempos de execução razoáveis.

Os problemas de empacotamento, em geral, buscam determinar o arranjo ótimo de unidades menores (itens) dentro de unidades maiores (caixas) satisfazendo determinadas restrições, ou seja, consiste em alocar uma coleção de itens, com tamanhos definidos, no menor número de compartimentos. A capacidade do compartimento é previamente definida, de forma que nenhum destes seja preenchido além de sua capacidade. A aplicação destes problemas é fundamental para o planejamento da produção de vários segmentos industriais, cujo objetivo é minimizar os efeitos negativos gerados por desperdício de materiais e igualmente relevantes no planejamento de operações logísticas, visando a minimização de espaços ociosos [Golden 1976]. A preocupação em utilizar mecanismos que busquem a proximidade com soluções ótimas é priorizar o melhor resultado em tempo hábil.

De acordo com Arenales (1993), a caracterização do problema, para determinação do tipo de problema, possui aspectos básicos a serem focados com respeito à estrutura, são eles:

- Dimensionalidade;
- Tipo de seleção dos objetos/itens;
- Características dos objetos/itens;
- Restrições associadas aos padrões;
- Objetivos;
- Status da informação e variabilidade dos dados;
- Métodos de solução.

As três primeiras características da estrutura lógica são consideradas fundamentais, e são responsáveis pelo esquema que permite relacionar problemas de empacotamento entre si. De uma forma geral, os problemas de empacotamento consistem em armazenar uma quantidade de itens em recipientes. Dependendo da natureza dos itens, estes podem ser ditos unidimensionais, bidimensionais ou tridimensionais. O problema proposto neste trabalho é de natureza unidimensional, ou seja, considera-se apenas uma dimensão, o volume do item, onde o somatório dos volumes dos itens contidos em uma caixa nunca é superior à capacidade da mesma.

Para cada uma das instâncias do problema é preciso definir:

- A capacidade c da caixa;
- Uma lista de n objetos, o;
- Um número N de caixas, k.

2.3. Algoritmo Genético

Os Algoritmos Genéticos são inspirados no processo biológico de evolução natural de Darwin. Aplicam-se à solução de problemas reais por meio da simulação dos processos naturais de evolução, onde os fatores ambientais são identificados e explorados buscando convergir para soluções ótimas ou bem próximas, operando com populações de indivíduos representados por cromossomos.

Os animais competem entre si por diversos recursos, por exemplo, comida e água, o que, em uma mesma espécie, possibilita a provável redução de descendentes, tendo, portanto, menor probabilidade de seus genes serem propagados ao longo de sucessivas gerações [Esmin 2005]. Desta forma, um algoritmo genético toma como princípio uma população de indivíduos (população inicial do problema), avalia cada um (aplicação da função objetivo), seleciona os melhores e efetua manipulações genéticas, como cruzamento e mutação (correspondente às perturbações) com o objetivo de criar uma nova população, podendo obter indivíduos melhorados e adaptados às características de seu meio ambiente [de Castro Silva and Soma].

Segundo Esmin (2005), o princípio básico do funcionamento do AG obedece um critério de seleção que faz com que, após muitas gerações, a população inicial de indivíduos gere indivíduos mais aptos. Nos últimos anos observa-se um crescimento no uso de AG para soluções otimizadas e adequadas para problemas específicos, por exemplo, problemas que buscam uma solução em um espaço extenso de candidatos e que se restringem em um tempo. Neste cenário, Algoritmos Genéticos são aplicados como um mecanismo adaptativo para solucionar estes problemas.

De acordo com Faria (2016), os Algoritmos Genéticos iniciam sua busca por melhores soluções a partir de um conjunto primário de soluções, denominado população inicial. Essas soluções são avaliadas em paralelo, permitindo a busca em vários elementos da população. Cada elemento da população é denominado indivíduo, que pode ser representado, por exemplo, por uma cadeia de bits (ver Figura 1). Em seguida, por meio dos operadores genéticos de cruzamento e mutação, é gerada uma nova população a partir da população inicial e, assim, uma nova geração é obtida. O cruzamento é aplicado para aqueles indivíduos com maior aptidão de reprodução, simulando assim o processo de evolução natural. Portanto, o GA implementa métodos de busca guiados pela aptidão, onde cada geração explora informações da geração anterior a fim reproduzir novas soluções que serão consideradas e novamente manipuladas para atingir melhores resultados.

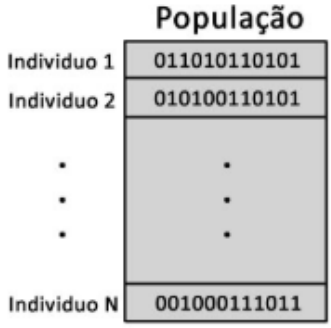

Figura 1. Exemplo de população GA por números binários

A Figura 2 apresenta o fluxo básico de execução do Algoritmo Genético. Após a identificação do conjunto primário pop_i, são selecionados dois indivíduos, que são comparados por meio de uma função objetivo previamente determinada. O indivíduo mais apto, ou seja, com melhor valor na função objetivo, segue adiante para a próxima geração. Esse processo é repetido até que as N posições de pop_i sejam contempladas pelos melhores indivíduos. Determina-se uma taxa de *crossover*, ou seja, a probabilidade de dois

indivíduos serem cruzados. Realiza-se a operação de cruzamento entre indivíduos com o intuito de gerar novos filhos, e melhorar o valor de adaptação. Após a realização do *crossover*, tem-se a operação de mutação [Goldberg and Holland 1988].

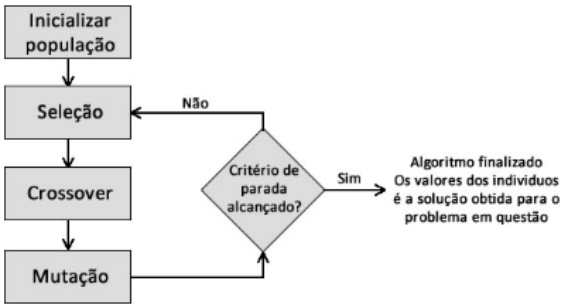

Figura 2. Fluxograma funcionamento do AG

Descrevendo os passos de funcionamento do AG, temos os seguintes conceitos:

- **População Inicial:** para que o algoritmo comece a ser executado é necessário a inserção de uma população inicial, ou seja, um conjunto de indivíduos. Neste trabalho, a população inicial é gerada de forma aleatória e representa possíveis soluções para o problema.

- **Seleção:** este operador determina quais indivíduos da população têm a maior probabilidade de serem escolhidos para o cruzamento, indivíduos pais. A seleção analisa os indivíduos mais aptos, que tenham maiores chances de sobreviver, e tem como objetivo principal copiar boas soluções [Faria 2016]. Uma das estratégias para selecionar os indivíduos pais é a seleção por torneio, onde ocorre a escolha de dois indivíduos de forma aleatória e o indivíduo pai é o que possui melhor valor de aptidão. Desse modo, realizam-se várias competições entre duas soluções e a melhor solução é copiada na lista de soluções que servirão para gerar os novos indivíduos [Deb 2001].

- **Crossover:** operador responsável por combinar as características dos pais, escolhidos na seleção, permitindo que as próximas gerações herdem essas características. Representa uma taxa do operador genético que predomina, por isso aplica-se uma taxa de *crossover* maior que a taxa de mutação [Cogo and Furtado 2001].

- **Mutação:** produz uma alteração aleatória em uma posição de um pequeno número de indivíduos, possibilitando uma maior variabilidade de material genético dentro da população. A mutação é a segunda maneira dos algoritmos genéticos explorarem o espaço de busca. Esta pequena alteração causada é para tentar impedir que o algoritmo tenha convergência muito rápida, evitando sua estabilização em regiões de mínimos locais [Ticona 2003].

Podem ocorrer situações em que o processo de *crossover* e mutação apresente algumas soluções inválidas. Para esses casos, é necessário um operador de reparo ou correção (ver Algorithm 1). Esse operador de reparo analisa se a solução é válida, ou seja, se a quantidade de itens alocados nas caixas não ultrapassa sua capacidade suportada.

Caso seja inválida, o operador de reparo retira um item da caixa e aloca esse item na primeira caixa com espaço disponível.

A execução do algoritmo proposto para o problema *Bin Packing* é apresentado através do seguinte pseudo-código:

Algorithm 1 ALGORITMO GENÉTICO

Seja $S(t)$ a população de indivíduos na geração t.
$t \leftarrow 0$

$inicializar$ S(t)
$avaliar$ S(t)

while *o critério de parada não for satisfeito (Avaliação)* **do**
    $t \leftarrow t + 1$ selecionar $S(t)$ a partir de $S(t-1)$;
    aplicar crossover sobre $S(t)$;
    aplicar mutação sobre $S(t)$;
    aplicar correção sobre $S(t)$;
    avaliar $S(t)$;
end

Dada uma população inicial para o problema abordado, os indivíduos pertencentes a esta população são submetidos a avaliações recorrentes a cada nova geração, além dos filhos gerados a partir dos cruzamentos. O processo de avaliação será executado para cada geração, formada de pais e filhos, selecionando os mais aptos para a próxima iteração, até atingir a condição de parada, que é dada por uma determinada taxa estabelecida no Algoritmo Genético proposto. Quando a taxa de avaliações é atingida, ou seja, a condição de parada é alcançada, o algoritmo avalia a população atual e seleciona a melhor saída para o problema aplicado.

3. Abordagem Proposta

3.1. Definição do Problema

Dada uma lista de instâncias, cada uma contendo a capacidade das caixas, quantidade de itens e o peso de cada um dos itens, deve-se minimizar o número de caixas utilizadas para alocar todos os itens. Para a solução do problema descrito, optou-se por uma abordagem utilizando a meta-heurística Algoritmo Genético. Entretanto, somente a utilização do algoritmo não satisfaz todas as restrições do problema, uma vez que pode resultar soluções em que o número de elementos alocados numa caixa ultrapasse a capacidade da mesma. Desse modo, foram implementadas rotinas adicionais para tratar soluções inviáveis, como o operador de reparo, abordado na subseção 2.3.

3.2. Instâncias do Problema

O algoritmo foi submetido à análise em instâncias de 50, 100, 200, 500, 1500 e 3000 itens, onde a capacidade das caixas variam entre 300 e 1000. A Figura 3 apresenta um exemplo de arquivo de entrada do algoritmo. A linha 1 mostra a quantidade de itens (50) e a capacidade da caixa (1000), a próxima linha apresenta a sequência dos 50 elementos que devem ser armazenados nas caixas. Ao todo, foram utilizadas 10 instâncias para realização do experimento.

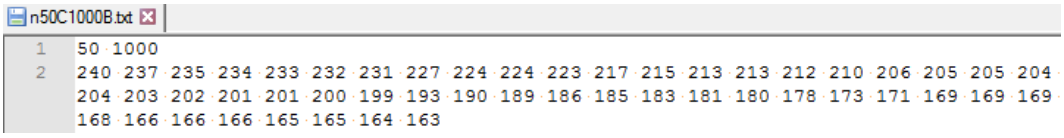

Figura 3. Exemplo de uma instância

3.3. Função Objetivo

Como dito, o Algoritmo Genético exige a criação de uma população. Cada indivíduo da população possui um valor de aptidão associado que identifica sua adaptabilidade ao ambiente. Desse modo, indivíduos com maior grau de aptidão são selecionados e possuem preferência para gerar novas populações. A função objetivo determina o valor de aptidão de cada indivíduo.

Neste trabalho, o problema de empacotamento foi modelado como um vetor, onde o índice consiste na posição do item e o valor armazenado corresponde ao seu volume. O algoritmo aloca os itens nas caixas de forma aleatória. A função gera um novo vetor que especifica o número da caixa que o item foi alocado, obedecendo a ordem de posição dos itens. A partir disso, é realizado um cálculo que obtém o somatório do espaço das sobra das caixas, denominado *fitness*. Cada indivíduo da população possui um valor de *fitness* e, através dele, é possível comparar uma solução boa com outra solução da população.

Considere o conjunto de objetos $O = \{o_1, o_2, ..., o_N\}$ disponíveis para serem selecionados para o conjunto de *bins* $K = \{k_1, k_2, ..., k_P\}$, onde N e P são respectivamente o número total de itens e *bins*. Nessa representação tem-se o vetor $X = \{x_1, x_2, ..., x_N\}$ onde $x_i \in \{0, 1, ..., P\}$, de modo que se $x_i = 0$ o objeto o_i está alocado para uma *bin* k_p, sendo $p = x_i$.

Figura 4. Representação da Solução

A Figura 4 exemplifica um cenário no qual os objetos o_2, o_4, o_6 e o_9 estão selecionados para a segunda *bin*, posteriormente os itens o_1, o_3 e o_8, para a décima *bin* e por fim, o_5, o_7 e o_{10} para a décima quinta *bin*.

3.3.1. Formulação Matemática

Segundo Vink (1997), o problema de empacotamento unidimensional pode ser formulado da seguinte maneira: dado P *bins* de capacidade c_i e N itens de peso, a solução ideal será alocar todos os itens em *bins*, de maneira que o tamanho total dos itens em uma caixa não exceda a capacidade c da *bin* e o número de caixas usadas seja o mínimo. Assim, o problema do empacotamento foi formulado da seguinte forma:

$$\begin{aligned} &\text{minimizar} \quad Fitness(X) = \sum_{i=1}^{P} c_i - \sum_{j=1}^{N} Volume(x_j),\ \forall x_j = i \\ &\text{sujeito a:} \quad \sum_{i=1}^{P} \sum_{j=1}^{N} Volume(x_j), \leq c_i \end{aligned} \tag{1}$$

onde $Volume(x_j)$ retorna o volume da *bin* k_p, sendo $p = x_j$. O valor de $Fitness(X)$ é menor a medida que os objetos são alocados para as mesmas *bins*.

Em relação a restrição do problema, cada objeto o_i tem um volume retornado pela função $Volume(x_j)$ e cada *bin* k_i tem uma capacidade total c_i. Assim, é necessário que a soma dos volumes de todos os objetos atribuídas a cada *bin* k_i não exceda a capacidade c_i definida para a bin i.

4. Testes e Resultados

4.1. Configuração do experimento

Os testes foram realizados em um computador com o processador Intel Core i5 e Memória RAM de 4Gb. Em relação à técnica de otimização, foi implementado a meta-heurística Algoritmo Genético utilizando o *Framework JMetal* [Durillo and Nebro 2011], escrito na linguagem de programação JAVA, e executado com as seguintes configurações:

- Tamanho da população: 100
- Taxa de *crossover*: 0.9
- Taxa de mutação: 0.01
- Taxa de avaliações no primeiro experimento: 1 milhão
- Taxa de avaliações no segundo experimento: 10 milhões

4.2. Performance do Algoritmo

Para verificar a performance do algoritmo e estabelecer as taxas de *crossover*, mutação e avaliação, apresentadas na seção anterior, foram realizados alguns testes na instância n100C1000A, a qual possui 100 itens e caixas com 1000 de capacidade. Cada instância foi testada 5 vezes em cada configuração, sendo considerado para tabulação de resultados (ver Tabela 1) apenas o melhor valor obtido em cada configuração, ou seja, a solução de menor *fitness* e, consequentemente, menor quantidade de *bins*.

Tabela 1. Calibração da instância: n100C1000A

Nome da Instância		n100C1000A		”Ótimo”:		12
Avaliação	Crossover	Mutação	População	Fitness	Bin	Tempo de Execução(s)
250000	0.9	0.01	100	8727	20	1,645
1000000	0.9	0.01	100	7727	19	5,530
10000000	**0.9**	**0.01**	**100**	**3727**	**15**	**54,866**
10000000	0.9	0.02	100	6727	18	54,442
10000000	0.5	0.01	100	4727	16	51,298

As manipulações dos valores foram embasadas em trabalhos conhecidos na literatura, para os operadores de *crossover*, mutação e quantidade de avaliações. Assim, para a realização dos testes nas demais instâncias, adotou-se a configuração que apresentou o melhor resultado.

Como demonstra a Tabela 1, a linha que representa o menor *fitness* e menor *bin* possui 10 milhões de avaliações, (90%) de *crossover* e (1%) de mutação. Preservadas essas taxas de *crossover* e mutação, os demais experimentos foram realizados nas 10 instâncias analisadas para duas taxas de avaliações diferentes, estabelecendo dois grupos de experimento. O grupo 1 tem taxa de avaliação de 1 milhão e o grupo 2, 10 milhões. Cada instância foi executada 5 vezes, e os resultados são apresentados na Tabela 2 (grupo 1) e Tabela 3 (grupo 2).

Tabela 2. Resultados Grupo 1 - 1.000.000 Avaliações

Instâncias Comparativas				Solução	Solução Obtida			Aproximação
N^o	Referência	n	C	"Ótima"	Inicial	tempo(s)	Melhor	Gap
1	n50C1000A	50	1000	18	19	3,064	19	1,05
2	n50C1000B	50	1000	10	12	3,059	11	1,10
3	n100C1000A	100	1000	12	19	6,359	17	1,41
4	n200C1000A	200	1000	67	75	11,990	73	1,08
Média das aproximações com resultados conhecidos ⇒								1,16
N^o	Outras	n	C	Inicial	Mínima	Média	Melhor	tempo(s)
5	n100C1000B	100	1000		29	28	28	5,604
6	n200C1000B	200	1000		53	51	49	10,592
7	n500C300	500	300		205	204	203	51,917
8	n500C1000	500	1000		156	151	147	27,438
9	n1500C1000	1500	1000		642	631	619	154,825
10	n3000C500	3000	500		1763	1742	1721	2569,147

Tabela 3. Resultados Grupo 2 - 10.000.000 Avaliações

Instâncias Comparativas				Solução	Solução Obtida			Aproximação
N^o	Referência	n	C	"Ótima"	Inicial	tempo(s)	Melhor	Gap
1	n50C1000A	50	1000	18	19	47,305	**19**	1,05
2	n50C1000B	50	1000	10	11	67,528	**10**	1,00
3	n100C1000A	100	1000	12	17	119,883	**15**	1,25
4	n200C1000A	200	1000	67	73	281,328	**72**	1,07
Média das aproximações com resultados conhecidos ⇒								1,09
N^o	Outras	n	C	Inicial	Mínima	Média	Melhor	tempo(s)
5	n100C1000B	100	1000		29	28	**28**	135,263
6	n200C1000B	200	1000		49	49	**48**	262,155
7	n500C300	500	300		203	201	**199**	864,819
8	n500C1000	500	1000		147	140	**132**	578,480
9	n1500C1000	1500	1000		619	583	**546**	3768,936
10	n3000C500	3000	500		1713	1690	**1659**	23502,187

Todas as instâncias submetidas às configurações do experimento do grupo 2 apresentaram um resultado melhor que as submetidas ao grupo 1, com um menor número de

caixas utilizadas, com exceção da primeira que apresentou o mesmo resultado. Entretanto, observa-se o aumento considerável do tempo médio de execução de cada instância. Das quatro instâncias que possuem o valor ótimo de caixas conhecido, o algoritmo atingiu apenas o da instância n50C1000B.

De modo geral, o algoritmo apresentou bons resultados para o problema de empacotamento. O tempo médio de execução é diretamente proporcional ao tamanho da instância, número de avaliações e taxa de configuração dos operadores. Analisando a instância maior, percebe-se que o tempo de execução ultrapassa 42 minutos no resultado do grupo 1 e quase 7h (6h52min) no grupo 2, alocando todos os itens em 1721 caixas no primeiro e 1659, no segundo. Comparando os resultados, o algoritmo demonstrou capacidade de otimizar os itens da instância do grupo 2 em 62 caixas a menos que o grupo 1. No entanto, com um tempo de execução bem maior.

4.3. Representação dos Resultados

A Figura 5 mostra a representação do melhor resultado obtido para a instância n50C1000A, submetida à configuração pertencente ao grupo 2, com condição de parada em 10 milhões de avaliações. Observa-se que o algoritmo exerce um esforço para o preenchimento total da capacidade da caixa. Entretanto, as últimas apresentam um espaço de sobra maior. Através da comparação dos resultados obtidos nos dois grupos de análise, pode-se afirmar que quanto maior o número de avaliações maior a probabilidade de obter um resultado que minimize o número de caixas necessárias. Porém, o tempo de execução cresce consideravelmente, dependendo do tamanho da instância.

Dados da instância:

- **Número de Itens**: 50
- **Pesos dos itens**: 392 392 391 390 390 388 386 382 381 380 380 380 375 375 375 374 373 372 370 364 360 360 359 355 346 345 343 341 332 320 317 317 314 313 311 308 307 305 303 296 294 290 283 282 280 274 273 272 269 267
- **Capacidade da Caixa**: 1000

Dados de configuração do algoritmo:

- **Tamanho da população**: 100
- **Avaliações**: 10.000.000 (função de parada)
- **Crossover**: 90%
- **Mutação**: 1%
- **Executado em**: 47305ms
- **Saída do algoritmo**: 18 3 0 2 1 16 4 14 17 5 7 9 6 8 12 13 13 17 11 11 12 18 10 14 15 16 5 15 9 10 6 10 1 15 2 7 3 8 4 7 3 4 2 0 6 1 8 9 0 5
- **Caixas utilizadas**: 19

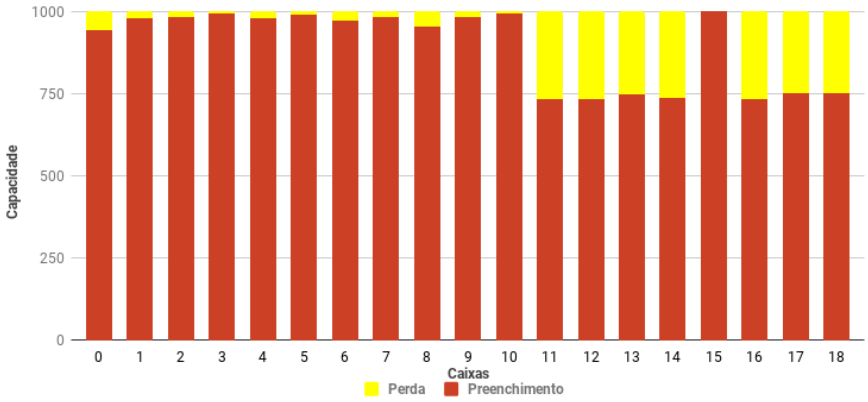

Figura 5. Preenchimento das Caixas para Instância n50C1000A

5. Conclusão

O Problema de Empacotamento é um problema da classe NP-Difícil e ao longo dos anos vários algoritmos foram desenvolvidos buscando soluções mais próximas do ótimo. Este trabalho apresentou uma discussão sobre o Algoritmo Genético e Problema de Empacotamento e implementou uma solução para minimizar o número de caixas necessárias para cada instância dada no estudo. Para tanto, foi fundamental contextualizar e caracterizar o problema.

Para a solução do Problema de Empacotamento tratado neste trabalho, implementou-se a meta-heurística Algoritmo Genético com a finalidade de retornar uma solução satisfatória para o problema proposto. Modificações nos valores dos operadores foram implementadas a fim de promover uma variação no espaço de busca. Além do algoritmo genético, foi implementado um operador de reparo que garante que as soluções geradas a partir de cada geração sejam válidas. Os resultados mostram que a meta-heurística AG em conjunto com as alterações propostas e os parâmetros utilizados, é um mecanismo eficiente para encontrar soluções próximas do ótimo para o problema descrito, e em tempo hábil.

Ciente de que este artigo alcançou o objetivo proposto, mas que a meta-heurística utilizada necessita de melhorias para minimizar o número de caixas e o otimizar o processo de alocação dos itens, propõe-se como trabalhos futuros, aprimorar o operador de reparo de forma que o processo de alocação priorize colocar os itens nas caixas que já estão em uso e que ainda possuem espaço. Caso não encontre disponibilidade, só então ele criará uma nova caixa.

Referências

Arenales, M. (1993). Uma teoria para o problema de corte. *Uma Teoria para o Problema de Corte*.

Arroyo, J. E. C. et al. (2002). Heurísticas e metaheurísticas para otimização combinatória multiobjetivo.

Brown, R. (1988). *Topology: a geometric account of general topology, homotopy types and the fundamental groupoid*. Halsted Press.

Cogo, A. P. and Furtado, J. C. (2001). Otimização do problema de corte unidimensional na indústria usando algoritmos genéticos. *Trabalho de conclusão de curso (Graduação em sistemas de informação). Centro Universitário Fransiscano*.

de Castro Silva, J. L. and Soma, N. Y. Um algoritmo genético aplicado ao problema de empacotamento de bins tridimensionais.

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons.

Durillo, J. J. and Nebro, A. J. (2011). jmetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, 42(10):760–771.

Esmin, A. A. A. (2005). *Estudo de aplicação do algoritmo de otimização por enxame de partícula na resolução de problemas de otimização ligados ao SEP. 2005*. PhD thesis, Tese (Doutorado em Engenharia Elétrica)-Universidade Federal de Itajubá, Minas Gerais.

Faria, A. O. (2016). Otimização do problema de corte e empacotamento unidimensional utilizando algoritmo genético. `http://repositorio.ufla.br/handle/1/5525`. (Accessed on 06/10/2018).

Garey, M. R. and Johnson, D. S. (1979). Computers and intractability: A guide to the theory of npcompleteness (series of books in the mathematical sciences), ed. *Computers and Intractability*, page 340.

George, J. A. and Robinson, D. F. (1980). A heuristic for packing boxes into a container. *Computers & Operations Research*, 7(3):147–156.

Goldberg, D. E. and Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine learning*, 3(2):95–99.

Golden, B. L. (1976). Approaches to the cutting stock problem. *AIIE transactions*, 8(2):265–274.

Morabito, R. and Arenalest, M. (1994). An and/or-graph approach to the container loading problem. *International Transactions in Operational Research*, 1(1):59–73.

Ticona, W. G. C. (2003). Aplicação de algoritmos genéticos multi-objetivo para alinhamento de seqüências biológicas. *Instituto de Ciências Matemáticas e Computação, Universidade de São Paulo, São Carlos, SP*.

Vink, M. (1997). Solving combinatorial problems using evolutionary algorithms. *Master's thesis, Leiden University*.