FACULDADES INTEGRADAS DE CARATINGA

FACULDADE DE CIÊNCIA DA COMPUTAÇÃO

APLICAÇÃO DE UM ALGORITMO GENÉTICO MODIFICADO AO PROBLEMA DE EMPACOTAMENTO UNIDIMENSIONAL

GILBERTO GOMES PACHECO

Caratinga

2012

Gilberto Gomes Pacheco

APLICAÇÃO DE UM ALGORITMO GENÉTICO MODIFICADO AO PROBLEMA DE EMPACOTAMENTO UNIDIMENSIONAL

Monografia apresentada ao Curso de Ciência da Computação das Faculdades Integradas de Caratinga como requisito parcial para obtenção do título de Bacharel em Ciência da Computação orientado pela Professora Msc. Míriam de Souza Monteiro.

Caratinga

2012

Gilberto Gomes Pacheco

APLICAÇÃO DE UM ALGORITMO GENÉTICO MODIFICADO AO PROBLEMA DE EMPACOTAMENTO UNIDIMENSIONAL

Monografia submetida à Comissão examinadora designada pelo Curso de Graduação em Ciência da Computação como requisito para obtenção do grau de Bacharel em Ciência da Computação.

_____

Prof. Msc. Míriam de Souza Monteiro
Faculdades Integradas de Caratinga

_____

Prof. Msc. Fabrícia Pires Souza Tiola
Faculdades Integradas de Caratinga

_____

Prof. Msc. Paulo Eustáquio dos Santos
Faculdades Integradas de Caratinga

AGRADECIMENTOS

Agradeço a Deus, aos familiares pelo apoio e incentivo. A meu amigo Wesley Gomes de Almeida por ter me ajudado na escolha do tema de minha monografia. A minha orientadora Miriam de Souza Monteiro por ter me ajudado nas diversas etapas deste trabalho. A todos os meus professores por terem me passado um pouco de seu conhecimento. Enfim, gostaria de agradecer a todos que de uma forma ou de outra me ajudaram a chegar aonde cheguei.

“A menos que modifiquemos a nossa maneira de pensar, não seremos capazes de resolver os problemas causados pela forma como nos acostumamos a ver o mundo”

Albert Einstein

RESUMO

Em otimização combinatória uma das classes de problemas clássicos e de grande aplicação prática é a classe de problemas de corte e empacotamento, tais problemas tem sido muito estudados e diversas soluções foram propostas para sua resolução, dar-se-á ênfase neste trabalho numa ramificação do problema de empacotamento, conhecido como “Problema de Empacotamento Unidimensional”. O problema consiste em: dados um ou mais objetos unidimensionais, chamados de recipientes com capacidade fixa, e vários objetos menores também unidimensionais, chamados de itens, empacotar todos os itens dentro dos recipientes, com o objetivo de maximizar o espaço alocado em cada recipiente de forma utilizar o menor número destes. Apesar da facilidade de definição do problema, sua otimização (obter o melhor arranjo segundo o objetivo descrito acima) consiste numa tarefa árdua, uma vez que se trada de um problema pertencente à classe NP-difícil. Portanto ainda não se conhece nenhum algoritmo exato que possa resolvê-lo em tempo hábil. Considerando que algoritmos exatos para resolução deste problema não são eficientes devido ao grande esforço computacional dependendo do tamanho da entrada de dados, uma estratégia consiste em aplicação de heurísticas que apesar de não garantirem uma solução ótima, produzem boas soluções em um tempo de execução admissível. O presente trabalho utiliza um algoritmo genético para a resolução do problema de empacotamento de *bins* unidimensionais, devido a sua escalabilidade a diversos problemas de otimização combinatória e bons resultados obtidos em tais problemas.

Palavras-Chave: Problema do Corte e Empacotamento, Algoritmo Genético Modificado, Problema do Empacotamento Unidimensional.

ABSTRACT

In combinatorial optimization one of the classes of classic and great practical application is the cutting and packing class problem, such problems have been widely studied and various solutions have been proposed for its resolution, it is given emphasis in this paper on one ramification of the packing problem, known as “One-dimensional Bin Packing Problem”. The problem consists as follow: given one or more one-dimensional objects, called *bins* of fixed capacity, and many smaller also one-dimensional objects, called *items*, to pack all the *items* into the *bins*, having the objective of maximizing the allocated space in each *bin*, so that the minimum number of *bins* is used. Despite the ease of defining the problem, the optimization (the best arrangement as described in the objective above) is a difficult task, since it is a problem that belongs to the NP-hard class. That way there are still no known efficient exact algorithm that can solve it in an acceptable time. Whereas exact algorithm for solving this problem are no effective, a strategy consists of applying heuristics that while not providing an optimal solution, produce good solutions in an acceptable run time. This paper makes use of a genetic algorithm for solving the one-dimensional bin packing problem, due to its scalability to various combinatorial optimization problems and good results in such problems.

Keywords: Cutting and Packing Problem, Modified Genetic Algorithm, One-Dimensional Bin Packing Problem.

LISTA DE SIGLAS

AGM – Algoritmo Genético Modificado.

BPP – *Bin Packing Problem.*

CSP – *Cutting Stock Problem.*

HD – *Hard Drive.*

HGGA – *Hybrid Grouping Genetic Algorithm.*

KP – *Knapsack Problem*.

NP – Não Determinístico Polinomial.

NPC – Não Determinístico Polinomial Completo.

P – Polinomial.

PCE – Problema de Corte e Empacotamento.

SBSBPP – Single Bin Sized Bin Packing Problem

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O problema de corte e empacotamento (PCE) envolve diversas situações práticas no dia-dia de diversas empresas e áreas industriais, tais como: metalurgia, siderurgia, marcenaria, informática, logística, entre outras. A obtenção de bons algoritmos para resolução do PCE traria uma economia considerável tanto em questão de matéria prima, o que poderia inclusive trazer benefícios ecológicos, quanto melhoria no rendimento dos processos em tais áreas, o que traria benefícios econômicos às empresas.

O PCE divide-se em vários tipos específicos, tais como: Problema de Corte de Estoque, Problema da Mochila, Problema de Empacotamento entre outros, cada um com sua particularidade. Neste trabalho dar-se-á ênfase no problema de empacotamento de *bins* unidimensionais, que segundo Noronha (2001), consiste em: dada uma lista de *n* itens unidimensionais com seus respectivos pesos, o objetivo será encontrar o menor número de *bins* de uma determinada capacidade necessários para empacotá-los. Este problema apesar de sua simples definição faz parte da classe de problemas NP-difícil (JOHNSON, 1986), sendo assim é pouco provável que se encontre um algoritmo exato eficiente que o resolva.

Devido a sua grande aplicação prática, diversos pesquisadores propuseram soluções para resolver o PCE, porém dada a sua complexidade e natureza altamente combinatória, outras estratégias além de algoritmos exatos (os quais oferecem o tempo proibitivo para obtenção de uma resposta para entradas relativamente grandes) são adotadas tais como: programação linear (CAVALI, 2004) e algoritmos genéticos (FARIA, 2006).

Dentre os métodos computacionais utilizados para se resolver problemas combinatórios complexos como PCE, destacam-se os algoritmos genéticos, que são algoritmos probabilísticos muito eficientes, que se adéquam facilmente a formulação de diversos problemas e oferecem bons resultados em tempo hábil de execução.

Dada a importância do PCE, o objetivo deste trabalho é implementar um algoritmo genético com algumas modificações nos operadores de seleção e cruzamento, porém mantendo a maior parte de sua estrutura clássica, aplicá-lo ao PCE e compará-lo às soluções obtidas por Falkenauer (1996), uma vez que este

também utiliza uma variação do algoritmo genético clássico para resolver este problema. Os dados de entrada para comparação da eficiência do algoritmo proposto serão detalhados no capítulo sobre os resultados obtidos e são os mesmos usados por Falkenauer (1996), o que permite uma comparação direta de eficiência entre as duas soluções.

2 REFERENCIAL TEÓRICO

Desde a invenção do primeiro computador digital, um avanço enorme tem ocorrido em relação à capacidade de processamento, armazenamento e transferência de dados. Tecnologias que antes estavam ao alcance apenas de grandes multinacionais e centros de pesquisa, hoje estão disponíveis em larga escala ao usuário comum (FONSECA FILHO, 2007).

Muitos problemas práticos do dia-dia puderam ser resolvidos com o auxílio de computadores, porém quando se trata de problemas altamente combinatórios, como o abordado neste trabalho, o avanço tecnológico esbarra na complexidade matemática inerente a tais problemas, o que em muitos casos, inviabiliza a obtenção de resultados ótimos (melhor resultado) devido ao tempo proibitivo para obtenção das respostas, mesmo utilizando os mais rápidos e poderosos computadores.

Os próximos tópicos abordam o tema da dificuldade de alguns problemas serem resolvidos em sua otimalidade, dentre os quais se dará ênfase ao problema de corte e empacotamento, e a estratégia adotada para contornar tal situação.

2.1 OTIMIZAÇÃO COMBINATÓRIA

Problemas de otimização combinatória fazem parte do campo de estudo da pesquisa operacional, que foi um modelo desenvolvido a partir da segunda guerra mundial com o objetivo de determinar a melhor utilização dos recursos militares da época (MONTEVECHI, 2000). Após o término da guerra, verificou-se que as técnicas desenvolvidas para fins militares se adequavam muito bem a problemas práticos do dia-dia e passaram a ser utilizadas por grandes organizações.

Tais problemas consistem em maximizar ou minimizar alguma determinada função objetivo, sujeitas às restrições inerentes ao problema estudado. Como exemplo, tem-se o problema de roteamento de veículos, no qual se tem um depósito onde saem caminhões carregados de produtos que devem ser entregues em algumas cidades, porém existe uma capacidade máxima que os caminhões

podem carregar em cada viagem. O objetivo é minimizar o gasto total (quilômetros rodados ou combustível gasto) para entregar todos os produtos respeitando a capacidade do caminhão (MIYAZAWA, 2012).

Ainda segundo Miyazawa (2012), problemas combinatórios como, o do caixeiro viajante, problema da mochila, problema da satisfatibilidade máxima, entre outros, surgem naturalmente em aplicações práticas tais como, projeto de redes de telecomunicação, de circuitos integrados, otimização logística e outros mais.

Entre os problemas tratados em otimização combinatória se encontra o PCE, o qual será apresentado em detalhes na seção 2.7.

2.2 ALGORITMOS

Nesta seção serão apresentadas definições importantes sobre algoritmos e problemas computáveis. Questões como: intratabilidade, algoritmos polinomiais, algoritmos exponenciais e a relação entre estas definições e o PCE serão explicadas.

2.2.1 Definição de Algoritmo

Algoritmos são sequências de ações executáveis com o objetivo de obter uma solução para um determinado tipo de problema (ZIVIANI, 1993). Ainda segundo Dijkstra (1971) citado por Ziviani (1993), “um algoritmo corresponde a uma descrição de comportamento, expresso em termos de um conjunto finito de ações”.
Em outras palavras pode-se definir algoritmo como um conjunto de etapas bem definidas que podem ser executados manual ou automaticamente de forma que dada uma entrada de dados, através de operações predefinidas, este possa devolver uma saída.

2.2.2 Complexidade de Algoritmos

Quando se pretende criar ou utilizar um algoritmo para alguma função específica, é sempre importante determinar o seu desempenho. Todo algoritmo tem o objetivo de executar uma determinada função, e para essa tarefa ele utiliza certa quantidade de recursos, tais como: memória, largura de banda e tempo de execução (ASCENCIO e ARAÚJO, 2010). Podem existir diversos algoritmos que resolvam um problema em particular, portanto, é necessário analisar qual deles é mais eficiente levando-se em consideração os recursos citados anteriormente.

Quando se analisa um algoritmo, são identificadas três situações de sua execução: melhor caso, pior caso e caso médio. Em um algoritmo de comparação sendo *n* o tamanho da entrada, o pior caso será quando o algoritmo tiver que comparar todos os *n* valores dessa entrada, o melhor caso será quando o valor a ser comparado estiver na primeira posição entre os *n* valores da entrada, e o caso médio será a média dos tempos de execução do algoritmo sobre todas as entradas de tamanho *n.* Nota-se que para valores pequenos de *n* mesmo um algoritmo pouco eficiente consumirá pouco tempo para resolver este problema, portanto o tempo gasto por um algoritmo está diretamente relacionado ao tamanho da entrada de dados, sendo assim, a análise de desempenho de um algoritmo é feita sobre tamanhos grandes de entrada. A essa análise dar-se o nome de eficiência assintótica, que se preocupa com o aumento do tempo de execução de um algoritmo à medida que o tamanho da entrada aumenta indefinidamente (ASCENCIO e ARAÚJO, 2010).

2.2.3 Notação *O*

Quando se analisa a execução de um algoritmo, procura-se saber qual é o seu comportamento para grandes valores de entrada. A notação *O* define um limite superior para o crescimento de uma função. Seja uma função *f(n)* e *g(n)*, dizemos que *f(n)* é *O(g(n))*, se *g(n)* cresce mais rapidamente que *f(n)* a partir de uma constante *c* e um valor de entrada n_0 (TOSCANI e VELOSO, 2002). Na Figura 1 -

Notação O, tem-se que a partir de um valor n_0 e uma constante *c*, o valor de *f(n)* sempre será menor ou igual ao produto de *c.g(n)*.

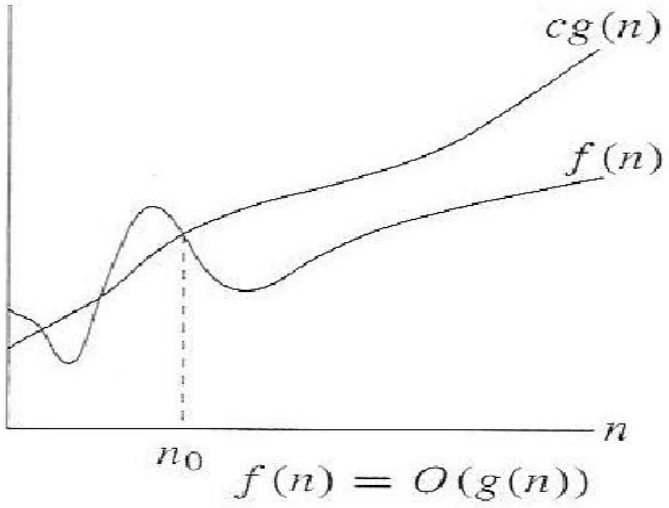

Figura 1 - Notação O.
Fonte: (TOSCANI e VELOSO, 2002 pag. 15).

Exemplificando: dada uma função *f(n) = 3n, g(n) = n²* e uma constante *c* = 1, com $n \in \mathbb{N}$, *g(n)* é um limite superior para *f(n)* a partir de n_0 = 3, pois *1.*(*g(n) = 3²*) ≥ *f(n)* = (*3.3*), para todo *n > c, e n* e *c* positivos.

Quando se afirma que um algoritmo tem seu tempo de execução definido por *O*(*n*²), significa existe uma função *f(n)* tal que para qualquer *n* a partir de um valor n_0, o tempo de execução para qualquer entrada tem um limite superior definido por *c.n²* (ASCENCIO e ARAÚJO, 2010).

2.2.4 Intratabilidade

Em ciência da computação algoritmos são considerados eficientes se conseguem resolver o problema proposto em tempo polinomial no tamanho da entrada (TOSCANI e VELOSO, 2002). Por exemplo: deseja-se verificar qual o maior número em um vetor de inteiros de tamanho *n*, não há outra possibilidade se não comparar todos os *n* valores desse vetor, tal algoritmo tem a complexidade dada em *O*(*n*). Algoritmos de complexidade $O(n)$, $O(n^k)$, $O(n \log n)$, sendo k um número real,

são chamados algoritmos polinomiais. Tais algoritmos oferecem soluções em tempo hábil de execução e os problemas resolvidos por estes algoritmos são chamados de problemas tratáveis (CORMEN et al., 2002).

Algoritmos não polinomiais como os de complexidade $O(n^k)$, $O(n!)$, são considerados não eficientes, pois exigem um tempo de execução proibitivo com o crescimento de n para grandes valores. Os problemas para os quais somente se conhecem tais algoritmos para sua resolução, são chamados de problemas intratáveis (TOSCANI e VELOSO, 2002).

2.3 TIPOS DE PROBLEMAS

Segundo Toscani e Veloso (2002) um problema é computável se ele pode ser resolvido mecanicamente (por um algoritmo), sendo que são três os tipos de problemas computáveis que se diferenciam pela colocação da pergunta: problema de decisão, problema de localização e problema de otimização.

- Problemas de decisão: São problemas para os quais se quer saber se a sua resposta é sim ou não, ou seja, o objetivo do algoritmo será comprovar uma resposta dada. Exemplo: dado um valor x contido em um vetor V de inteiros de tamanho n. Deseja-se saber, x é o maior valor dentre todos os n valores de V ?
- Problemas de localização: Neste caso, deseja-se encontrar uma resposta para um problema dado. Exemplo: em um grafo não orientado G e vértices u e v, encontre um caminho que saia de u e chegue v.
- Problemas de otimização: O interesse nessa classe de problemas é não somente localizar uma solução, mas sim uma solução ótima. Exemplo: No mesmo grafo não orientado G citado acima, com seus vértices u e v, encontre o menor caminho que saia de u e chegue até v.

2.4 A QUESTÃO P x NP

A questão na verdade é a seguinte $P = NP$? Sendo P (Polinomial) a classe de problemas para quais se conhece um algoritmo polinomial que os resolva, e NP (Não Determinista Polinomial) a classe de problemas em que se conhece um algoritmo polinomial para comprovar suas respostas, a classe P é igual a NP? Ou seja, para todo problema em que existe um algoritmo polinomial que comprove sua resposta também existe um algoritmo polinomial que o resolva? Certamente $P \subseteq NP$, pois se existe um algoritmo polinomial que o resolva (problema de localização), também existe um algoritmo polinomial que comprove sua resposta (problema de decisão), mas o contrário não é tão fácil de se provar (CORMEN et al., 2002).

A questão $P \times NP$ é um dos maiores e mais estudados problemas em ciência da computação e que ainda permanece sem resposta. Faz parte dos seis[1] problemas matemáticos ainda em aberto, ou seja, que não se conhece uma resposta. Há inclusive um prêmio de um milhão de dólares oferecido pelo *Clay Mathematics Institute*[2] para quem prove sua resolução (HARDESTY, 2009).

2.5 NP-COMPLETO

Um problema é NP-completo (NPC) se é não-polinomial e é tão difícil de resolver quanto qualquer outro problema em NP (CORMEN et al., 2002).

A característica mais interessante dos problemas NPC é que se existir um algoritmo que resolva um problema comprovadamente NPC em tempo polinomial então todos os problemas pertencentes a essa classe também podem ser resolvidos por algoritmos polinomiais. Porém segundo Cormen et al. (2002, p. 774):

---

[1] Em 24 de maio de 2000, foram divulgados os sete problemas ainda não resolvidos para os quais são oferecidos um prêmio de U$ 1 milhão de dólares pelo Clay Mathematics Institute, porém recentemente, em 18 de março de 2010, foi anunciado que um dos problemas (Conjectura de Poincaré) foi resolvido pelo Dr. Grigoriy Perelman de São Petersburgo na Rússia.

[2] Disponível em http://www.claymath.org/millennium/.

> A maioria dos teóricos da ciência da computação acredita que os problemas NP-completos são intratáveis, pois dada a ampla faixa de problemas NP-completos que foram estudados até hoje, sem qualquer progresso em direção a uma solução de tempo polinomial, seria verdadeiramente espantoso se todos eles pudessem ser resolvidos em tempo polinomial (CORMEN et al., 2002, p. 774).

2.6 NP-DIFÍCIL

A maior parte dos problemas NP - difíceis estão relacionados a problemas de otimização (MORAIS, 2010). A esta classe de problemas pertence o PCE. Outros problemas que fazem parte dessa classe são: problema do caixeiro viajante (PCV), problema da satisfatibilidade máxima, problema de parada (*Halting Problem)* entre outros (GRONER, 2006).

Todos os problemas dessa classe são ditos ao menos tão difíceis quanto os problemas mais difíceis de NP e não é possível resolvê-los na otimalidade em tempo polinomial, a menos que $P = NP$, sendo assim um algoritmo polinomial que encontre uma solução ótima para um problema NP-difícil implicaria que $P = NP$ (MORAIS, 2010).

2.7 O PROBLEMA DE CORTE E EMPACOTAMENTO

Um dos mais conhecidos e mais estudados problemas de otimização combinatória trata-se do PCE (Problema de Corte e Empacotamento).

O PCE, genericamente, consiste em cortar unidades maiores (objetos) em unidades menores (itens), ou empacotar unidades menores (itens) em unidades maiores (objetos) otimizando um certo objetivo (MORABITO, 1994).

Esse problema aparece em diversas situações práticas cotidianas como: corte de chapas de aço industriais, armazenamento de arquivos em CD’s e HD’s, carregamento de veículos, alocação de comerciais de TV, carregamento de contêineres, dentre outros similares (MORABITO, 1994).

Apesar do problema de cortar objetos maiores em itens menores (corte) ser diferente conceitualmente de empacotar itens em contêineres (empacotamento), matematicamente suas formulações são idênticas e por isso são estudados em conjunto.

Geometricamente o PCE pode ser dividido em uni, bi, e tridimensional, onde respectivamente são características intrínsecas: comprimento, área e volume dos objetos a serem cortados ou empacotados (MIYAZAWA, 2012). A Figura 2 representa um empacotamento de itens contendo uma, duas e três dimensões em objetos, também uni, bi e tridimensionais.

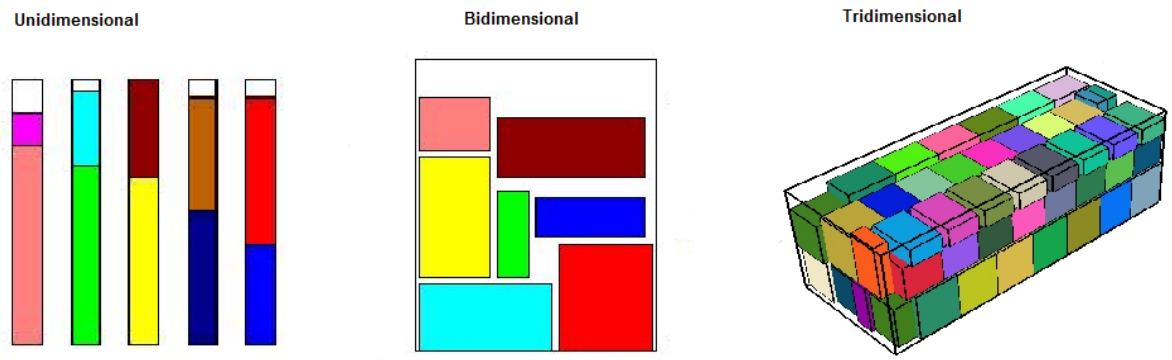

Figura 2 - Empacotamento de itens uni, bi e tridimensionais.
Fonte: (MIYAZAWA, 2012).

O PCE abrange uma grande quantidade de problemas mais específicos tais como: problema do empacotamento de itens (*BPP - Bin Packing Problem),* problema da mochila *(*KP - *Knapsack Problem),* problema do corte de estoque *(CSP - Cutting Stock Problem),* entre outros (HEIS, 2007). Uma breve descrição dos problemas citados é dada a seguir:

- Problema do empacotamento de *bins* (BPP): Nesse problema tem-se um número ilimitado de objetos que conterão os itens a serem empacotados. O objetivo é empacotar todos os itens usando o menor número possível de objetos.
- Problema da mochila (KP): O problema consiste em: dado um conjunto de itens com seus respectivos valores e dimensões, e um único objeto

de capacidade fixa (mochila), encontrar um arranjo tal de forma a maximizar o valor do objeto.

- Problema do corte de estoque (CSP): considerando um conjunto de itens grandes que devam ser cortados em itens menores e de tamanhos variados, o objetivo deste problema consiste em minimizar a perda de materiais que não possam ser mais utilizados, após a aplicação dos cortes.

Devido ao grande número de problemas específicos variantes do PCE, alguns pesquisadores propuseram formas de classificá-los, dentre os quais se destacam Dyckhoff (1990) e Wäscher et al. (2007). A classificação do problema estudado neste trabalho se baseia na tipologia deste último.

Segundo Wäscher et al. (2007) o problema apresentado nesse trabalho é conhecido como: **Problema de Empacotamento de *Bins* de Tamanho Único**, cuja nomenclatura seguindo sua tipologia é descrita como 1D-SBSBPP, o que corresponde à:

- 1D: Unidimensional
- SBSBPP: *Single Bin Sized Bin Packing Problem* (Problema de empacotamento de *bins* de tamanho único).

Como exemplo prático deste tipo específico de problema temos uma aplicação direta em informática, que se trata de alocar arquivos de diversos tamanhos em um menor número possível de setores de um HD ou menor número de CD's, otimizando o espaço alocado.

2.7.1 Formulação Matemática

Segundo Xavier (2006), no problema de empacotamento tem-se uma lista de itens $L = (a_1,...,a_m)$ como entrada, cada item com tamanho $s(a_i)$, e um número B que indica o tamanho de um recipiente. Se para todo item $a_i \in L$, vale que $s(a_i) \leq B$.

O problema consiste em empacotar todos os itens de L no menor número possível de recipientes, ou seja, deve-se achar uma partição $P_1,...,P_q$ de L de forma que *q* seja mínimo e $\sum a_i \in P_j s(a_i) \leq B$, para cada P_j.

Devido à sua grande aplicabilidade prática, diversos pesquisadores estudaram e propuseram soluções para resolvê-lo. Porém sabe-se que o PCE pertence à classe de problemas NP-Difícil (JOHNSON, 1986), o que pode impossibilitar a criação de um algoritmo que o resolva em tempo polinomial, sendo assim, faz-se necessária outra abordagem para sua resolução.

Quando um problema é difícil de ser tratado (não se conhece um algoritmo polinomial para resolvê-lo), costuma-se usar as chamadas heurísticas, que são algoritmos que procuram obter uma solução viável e de boa qualidade, que embora não garanta uma solução ótima, tem baixo custo computacional.

2.8 ALGORITMOS GENÉTICOS

Nesta seção apresenta-se o conceito de algoritmos genéticos, seus fundamentos na biologia e genética, sua estrutura clássica e aplicação em ciência da computação. Inicialmente será dada uma breve introdução aos estudos de Darwin e o desenvolvimento da genética como prelúdio para a criação dos algoritmos genéticos.

2.8.1 Teoria Darwinista

Em meados do século XIX, o naturalista inglês Charles Darwin iniciou uma viagem a bordo do navio HMS Beagle a diversas regiões do globo onde pode observar uma interessante relação entre a diversidade da vida animal e suas características específicas dependendo da região geográfica onde viviam (PAULINO 2003). Durante suas observações ele percebeu que a vida animal de alguma forma se adaptava ao seu habitat. Apesar de não saber quais os

mecanismos envolvidos na modificação dos indivíduos para melhor se adaptarem ao meio onde viviam, Darwin compreendeu que os indivíduos evoluíam de acordo com sua necessidade de sobrevivência.

2.8.2 Seleção Natural

Segundo Paulino (2003, p. 454) “Darwin considerou que certas características poderiam contribuir para a sobrevivência de certos indivíduos num determinado ambiente, constituindo-se, portanto, como variações favoráveis”. Ao passo que indivíduos que possuíssem características desfavoráveis em relação ao seu ambiente teriam dificuldade em manter a sobrevivência de sua espécie.

Neste ponto o ambiente exerce o papel de seletor das espécies mais aptas e consequentemente estas geram descendentes em maior número e com maiores chances de sobrevivência. Ao mesmo passo que todas as espécies estão sujeitas às restrições de seu ambiente e “evoluem” desenvolvendo características que as possibilitem gerar descendentes mais adaptados a este ambiente.

2.8.3 Genética Biológica

A genética é o estudo da herança biológica perpetuada entre os seres vivos, de geração para geração. Seu desenvolvimento teve início com as pesquisas do austríaco Gregor Mendel, o qual é considerando o pai da genética. Mendel utilizou ervilhas como objeto de seus experimentos, onde ele cruzava diferentes espécies dessas plantas e verificou que seus descendentes herdavam as características de seus progenitores (TAMARIN, 2001).

Com a descoberta dos cromossomos no final do século XIX, os geneticistas puderam entender como o processo de passagem das características vitais dos seres vivos se davam de uma geração para outra.

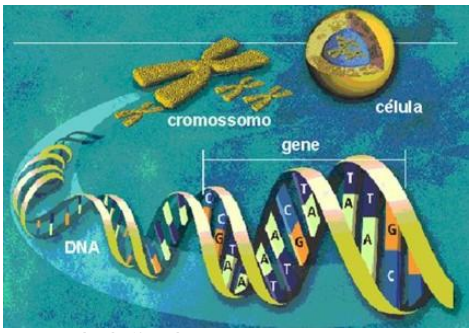

Figura 3 - Constituição de um cromossomo em uma célula.

Fonte: (ROCHA, 2012).

Como se pode notar pela Figura 3, o cromossomo é composto pelo DNA que é a base do material genético dos seres vivos, e este é composto pelos genes, que são a menor unidade de hereditariedade (TAMARIN, 2001). Na reprodução, os cromossomos dos pais se intercalam para dar origem a um novo cromossomo resultante da recombinação de genes entre eles, a Figura 4 representa tal fenômeno.

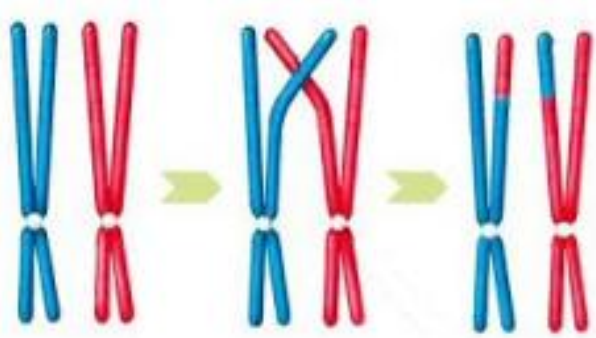

Figura 4 - Recombinação de dois cromossomos dando origem a um novo cromossomo.

Fonte: (OLIVEIRA, 2008).

2.8.4 Aplicação em Computação

Os algoritmos genéticos fazem parte da chamada computação evolutiva e provêem da aplicação da teoria darwiniana sobre seleção natural e o funcionamento da hereditariedade passada pelos cromossomos para resolução de problemas computacionais.

Proposto inicialmente por John Holland (1975) e seus alunos na universidade de Michigan – Estados Unidos. São algoritmos probabilísticos que fornecem um mecanismo de busca paralela e adaptativa baseado na sobrevivência do mais forte e sua reprodução (SOARES, 1997).

Como todo algoritmo não exato, os algoritmos genéticos não garantem a melhor solução para os problemas combinatórios, porém geram boas soluções em tempo admissível de execução.

2.8.5 Estrutura dos Algoritmos Genéticos

Assimilando sua analogia com os processos de evolução biológica, os algoritmos genéticos constroem um ambiente onde os indivíduos estão em constante evolução e adaptação aos requisitos do problema, ou seja, evoluem iterativamente para as melhores soluções. Os “indivíduos” que constituem a população dos algoritmos genéticos nada mais são que a representação das repostas para um dado problema, codificados segundo a necessidade do problema em questão (SOARES, 1997).

2.8.6 Codificação dos Indivíduos de Um Algoritmo Genético

Cada indivíduo possui um conjunto de características que o define, chamado de cromossomo (em analogia biológica). Sua representação usualmente é formada por uma cadeia de caracteres formando um vetor de tamanho definido. Os caracteres são as representações dos *genes* desse indivíduo (SOARES, 1997).

As representações mais utilizadas para os genes são: a binária e a decimal. A Figura 5 apresenta essas duas formas de representação.

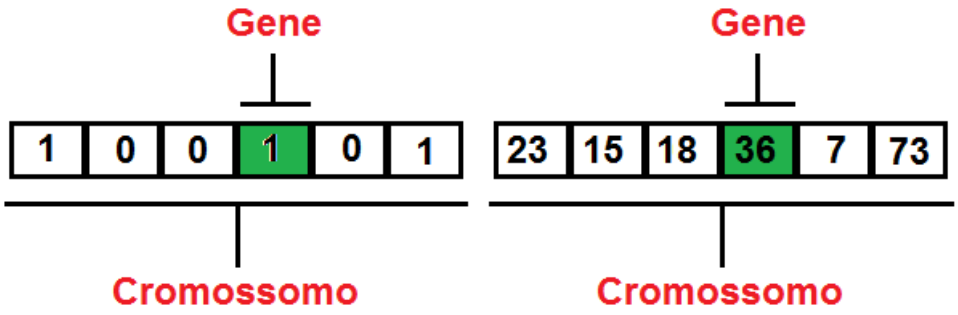

Figura 5 - Representação binária e decimal de um cromossomo em um algoritmo genético.

Fonte: Próprio Autor.

Abaixo se apresenta um pseudocódigo dos algoritmos genéticos proposto por Holland (1975):

Algoritmo Genético Clássico

1: P ← População Inicial;
2: Enquanto condição de parada não satisfeita faça:
3: P *t* ← Seleção(P);
4: P ← Cruzamentos (P *t*);
5: P ← Mutações(P);
6: Fim enquanto;
7: Solução Melhor indivíduo(P);

2.8.7 Detalhes da Estrutura Clássica dos Algoritmos Genéticos

A seguir, cada uma das etapas presentes na descrição do algoritmo genético proposto por Holland (1975) será explicada em detalhes.

- **População Inicial**

É o conjunto de indivíduos gerados de forma aleatória no início do algoritmo, que representam soluções para o problema, sendo que

posteriormente sofrerão modificações devido aos operadores de seleção, cruzamento e mutação, (FARIA, 2006).

- **Condição de Parada**

É o critério utilizado para parada do algoritmo, geralmente são utilizados os seguintes critérios: número de iterações fixas, número de iterações sem melhora da solução e tempo de execução (SOARES, 1997).

- **Seleção**

É o processo pelo qual os indivíduos da população atual serão escolhidos para dar origem a nova população, considerando seu grau de aptidão ao problema a ser resolvido, esses indivíduos são chamados de pais dos indivíduos presentes na geração seguinte (SOARES, 1997). Os métodos mais utilizados para se fazer a seleção de indivíduos em um algoritmo genético são apresentados a seguir:

  - **Roleta**

No método de seleção por roleta, cada indivíduo na população tem sua probabilidade de seleção calculada, seguindo o seguinte critério:

$$p_i = \frac{f(x_i)}{\sum_{k=1}^{N} f(x_k)}$$

Onde:

p_i: Probabilidade do indivíduo “i” ser escolhido.

$f(x_i)$: Valor da aptidão do indivíduo “i” ao problema a ser resolvido.

$\sum_{k=1}^{N} f(x_k)$: Somatório das aptidões dos outros indivíduos na população.

Supondo que $f(x_i)$ tenha o valor 10 e $\sum_{k=1}^{N} f(x_i)$ tenha o valor 50, a probabilidade do indivíduo *i* ser escolhido será 10 ÷ 50 = 0,2 ou 20%.

A partir do cálculo da probabilidade de cada individuo presente na população, monta-se uma roleta onde cada fatia da mesma é divida entre os indivíduos da população, alocando-se o tamanho da fatia a estes indivíduos conforme sua probabilidade de seleção calculada anteriormente. A Figura 6 apresenta uma roleta com a distribuição de probabilidade de escolha para quatro indivíduos.

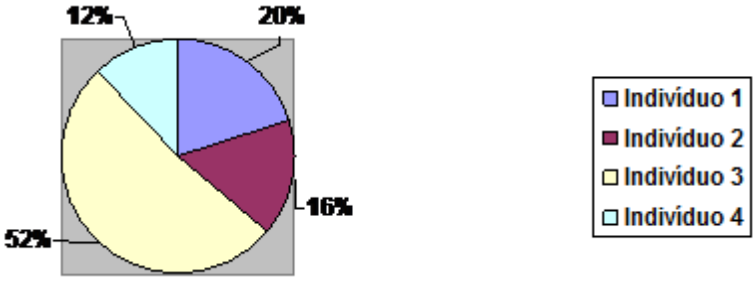

Figura 6 - Representação da probabilidade de seleção para quatro indivíduos.
Fonte: Próprio Autor.

Feita a distribuição das probabilidades, gira-se a roleta para a escolha dos indivíduos que irão cruzar-se para gerar a nova população do algoritmo genético.

- **Torneio**

Neste método escolhem-se *n* indivíduos ao acaso, os quais participaram de sucessivas disputas para determinar o melhor entre eles, este então é selecionado para fazer o cruzamento (LINDEN, 2008).

- **Cruzamento**

É o momento onde os indivíduos escolhidos na etapa anterior trocam seu material genético para formar novos indivíduos na população seguinte. Existem diversas formas de fazer o cruzamento (SOARES, 1997). As mais usuais serão detalhadas a seguir:

- **Cruzamento um Ponto**

Neste tipo de cruzamento, escolhe-se um ponto ao longo dos cromossomos dos indivíduos, que será usado para identificar o ponto de parada da cópia do primeiro indivíduo e o ponto de início do segundo. A Figura 7 apresenta um esquema de cruzamento um ponto.

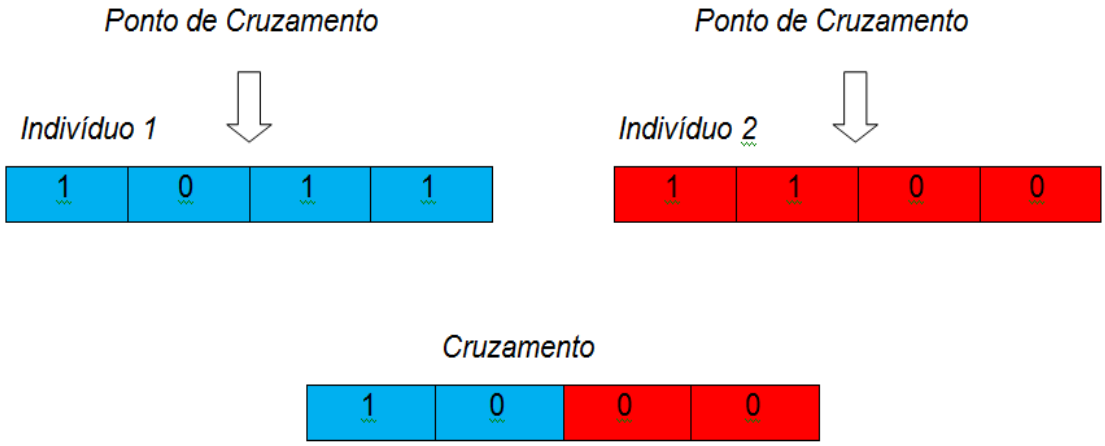

Figura 7 - Representação do cruzamento um ponto.

Fonte: Próprio Autor.

- **Cruzamento Multiponto**

Ao contrário do cruzamento de um ponto, no cruzamento multiponto, o cromossomo do individuo escolhido para fazer o cruzamento é divido em mais de uma posição como se destaca na Figura 8:

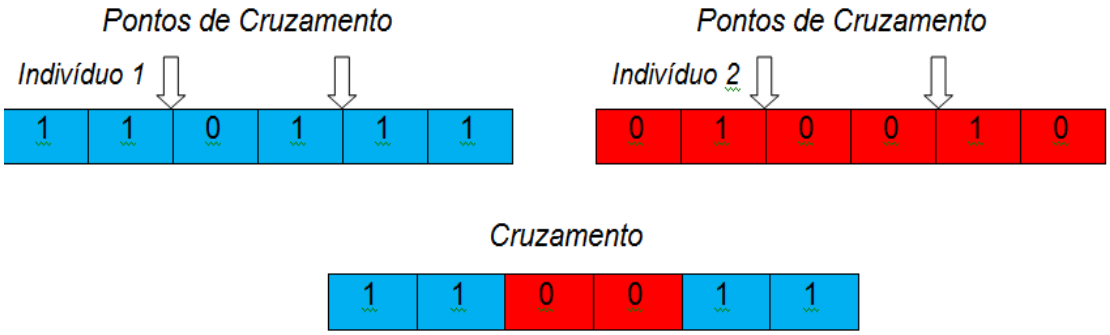

Figura 8 - Representação do cruzamento multi ponto.

Fonte: Próprio Autor.

- **Cruzamento Uniforme**

No cruzamento uniforme, cada gene do individuo gerado é composto pela intercalação dos genes de seus pais como visto na Figura 9:

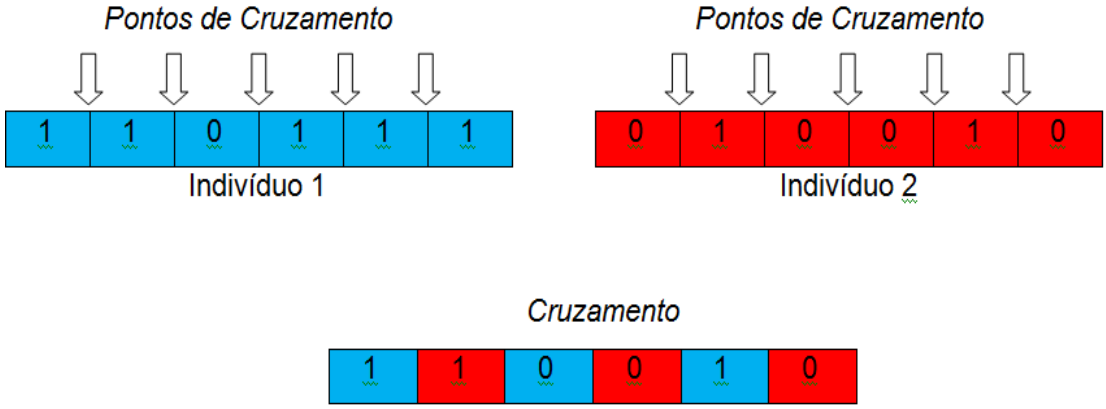

Figura 9 - Representação do cruzamento uniforme.

Fonte: Próprio Autor.

- **Mutação**

Assim como ocorre nos processos biológicos, os indivíduos presentes na população podem sofrer mutação, que são alterações em seu material genético não contido nos cromossomos dos seus pais. A mutação tem como objetivo fazer com que o algoritmo genético consiga escapar de ótimos locais (são resultados que parecem ser os melhores considerando somente uma

parte do espaço de soluções, mas que na verdade não são os melhores resultados entre todos existentes dentro do espaço total de soluções) e assim cobrir uma área maior no espaço de soluções (LINDEN, 2008).

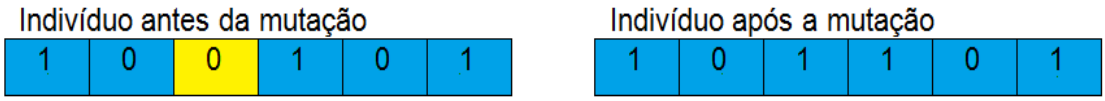

Figura 10 - Representação da mutação no terceiro gene do cromossomo de um indivíduo da população de um algoritmo genético.
Fonte: Próprio Autor.

O indivíduo representado na Figura 10 sofreu mutação no seu terceiro gene, como está representado pela codificação binária, este gene mudou de 0 para 1.

- **Repetição do Laço**

Após todas estas etapas serem concluídas, executa-se novamente a condição de parada, e caso seja satisfeita segue para o passo seguinte, caso contrário repetem-se novamente as etapas descritas acima.

- **Solução do Melhor Indivíduo**

Esta é a etapa final do algoritmo genético, é a etapa que devolve o melhor indivíduo (a melhor solução encontrada) nas sucessivas gerações criadas pelo algoritmo e que melhor se adaptou aos requisitos do problema. Em outras palavras, é a resposta final para o problema tratado.

Estas são as etapas clássicas de um algoritmo genético que serão usadas no decorrer do trabalho, modificando-se somente os operadores de seleção e cruzamento para se testar uma nova abordagem apresentada no próximo capítulo.

3 METODOLOGIA

O presente trabalho propõe um algoritmo genético com algumas alterações em sua estrutura clássica, aplicá-lo ao problema do empacotamento unidimensional, descrito na seção 2.7, com o objetivo de verificar sua eficácia para a resolução deste problema e compará-lo às soluções obtidas por Falkenauer (1996).

3.1 CODIFICAÇÃO DOS INDIVÍDUOS

Para que um problema possa ser resolvido por um algoritmo genético, ele deve estar codificado em uma estrutura de dados que possa ser reconhecida e manipulada por este algoritmo. Como já apresentado na seção 2.8.6, as formas mais utilizadas para codificação do problema em indivíduos que farão parte da população dos Algoritmos Genéticos, são: binária e decimal.

A codificação utilizada neste trabalho é a decimal, escolhida por sua relação direta com as características dos dados utilizados no problema do empacotamento unidimensional. A seguir será apresentado um problema fictício relacionado ao empacotamento unidimensional para fins de elucidação da codificação utilizada neste trabalho.

A forma de gravação de arquivos em um HD (Disco Rígido) utiliza um esquema conhecido como geometria de discos. Onde o disco é “dividido” em trilhas e setores (ALECRIM, 2012), como representado na Figura 11.

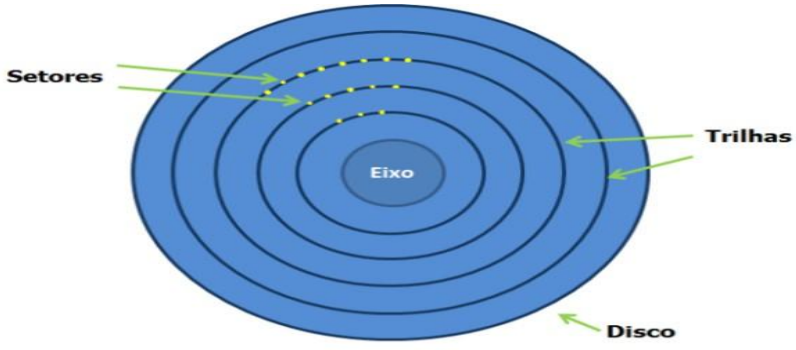

Figura 11 - Geometria de um disco rígido.

Fonte: (ALECRIM, 2012).

As informações são gravadas nos setores contidos nas trilhas, sendo que cada setor tem um tamanho fixo de 512 bytes. Pode-se imaginar cada setor como sendo uma “caixa” que conterá os arquivos “itens” a serem gravados no disco, como ilustra a Figura 12.

Figura 12 - Representação de um empacotamento de arquivos em setores de um disco rígido.
Fonte: Próprio Autor.

Percebe-se que quanto melhor o aproveitamento de cada setor (mais próximo de sua capacidade total), menor será o número de setores utilizados, o que ocasiona não somente aumento de espaço em disco, como também menor fragmentação do mesmo, o que consequentemente resulta em maior rapidez e desempenho do sistema.

Atribuindo-se os valores: 6, 4, 1, 7, 4, 9, 5, 4, 7, 3 bytes aos arquivos da esquerda para a direita e o valor 11 bytes para a capacidade do setor como demonstrado na Figura 12, a seguinte estrutura de dados, modelada em forma de um indivíduo de um algoritmo genético, foi usada para representar o empacotamento também presente na Figura 12.

Figura 13 - Individuo que representa o empacotamento da Figura 12.
Fonte: Próprio Autor.

No indivíduo representado pela Figura 13, cada gene de seu cromossomo com o valor diferente de zero, representa um arquivo alocado em um setor, o valor zero indica que foi utilizado um novo setor, pois o arquivo subsequente não caberia no setor anterior (lembrando que nessa situação cada setor possui 11 unidades de capacidade), portanto, o número de zeros indica o número de setores utilizados com exceção do início do cromossomo, pois os primeiros arquivos já estão alocados em um setor sem a utilização do valor zero. Sendo assim, o indivíduo acima utilizou cinco setores (quatro zeros mais o primeiro setor). O valor -1 indica final de empacotamento.

Pode-se notar que o tamanho total do cromossomo do indivíduo acima é igual ao dobro do número de arquivos a serem alocados mais um, pois no pior caso, será utilizado um setor para cada arquivo e, portanto, será necessário o dobro de genes mais um gene indicador de fim de empacotamento para codificar esta estrutura.

3.2 POPULAÇÃO INICIAL

A população inicial no algoritmo genético proposto neste trabalho é gerada através do uso de uma função randômica que, depois de lido o arquivo com os dados do problema, escolhe ao acaso os itens presente neste arquivo e os aloca aos genes dos indivíduos de forma sequencial, este processo é realizado até que se formem todos os indivíduos da população.

Foram definidos 100 indivíduos para representar a população do algoritmo genético proposto, pelo fato de ser um número bastante utilizado em outros trabalhos e ter melhor resultado em termos de tempo de execução neste trabalho.

3.3 ESTRUTURA CLÁSSICA VS ESTRUTURA PROPOSTA

Como já descrito na seção 2.8.6, um algoritmo genético clássico, possui as seguintes etapas:

1: P ← População Inicial;
2: Enquanto condição de parada não satisfeita faça:
3: P *t* ← Seleção(P);
4: P ← Cruzamentos (P *t*);
5: P ← Mutações(P);
6: Fim enquanto;
7: Solução Melhor indivíduo(P);

O algoritmo genético proposto neste trabalho também se baseia nas etapas descritas acima, porém, as etapas número 3 (Seleção) e 4 (Cruzamento) sofreram modificações para se testar a eficácia de uma outra abordagem que difere da estrutura clássica dos Algoritmos Genéticos.

3.3.1 Seleção (Estrutura Clássica)

Segundo Soares (1997), “O processo de seleção/reprodução é responsável pela escolha dos indivíduos que serão submetidos às operações genéticas como cruzamento e mutação”.

Independente do método de seleção utilizado em um algoritmo genético, todos se baseiam na escolha de dois indivíduos obtidos da população atual, que posteriormente se cruzarão e formarão outros indivíduos que farão parte da população seguinte. O processo de escolha de tais indivíduos usualmente é feito utilizando os métodos de roleta ou torneio apresentados na seção 2.8.7.

3.3.2 Seleção (Estrutura Proposta)

Na estrutura proposta, diferentemente da estrutura clássica, o modo como os indivíduos serão escolhidos para realizar o cruzamento, não utiliza nenhum dos métodos de seleção comumente utilizados como a roleta ou o torneio.

A forma escolhida para realizar a seleção se baseia em um princípio aplicado em agropecuária, que tem como objetivo selecionar o melhor indivíduo (animal ou planta) para ser o reprodutor do restante da população visando o melhoramento e homogeneidade do material genético do rebanho ou da plantação. Seguindo esta linha de raciocínio para escolha do(s) indivíduo(s), o método utilizado para seleção é simplesmente obter o(s) indivíduo(s) mais apto(s) em relação à adequação ao seu ambiente, ou seja, o indivíduo que possui o melhor material genético.

Neste trabalho, será escolhido apenas um indivíduo (o melhor segundo sua adaptação ao seu meio ambiente) para fazer o cruzamento com todos os outros indivíduos da população, o que dará origem à próxima geração da população.

Como apresentado na seção 3.1 a forma de avaliar a adequação do indivíduo ao seu meio, que neste caso se refere a utilizar o menor número de “caixas” alocadas consiste na contagem do número de zeros presentes em seu cromossomo. Portanto, o indivíduo que tiver o menor número de zeros em seu cromossomo será o indivíduo mais apto da população.

3.3.3 Cruzamento (Estrutura Clássica)

Os métodos mais comumente utilizados para se fazer o cruzamento em um algoritmo genético, são os cruzamentos: um ponto, multiponto e uniforme. Tais métodos foram apresentados na seção 2.8.7.

3.3.4 Cruzamento (Estrutura Proposta)

A forma escolhida para fazer o cruzamento neste trabalho difere abruptamente da forma clássica. Nenhuma das formas mencionadas anteriormente será utilizada no algoritmo proposto.

O cruzamento se dará entre o melhor indivíduo da geração atual da população do algoritmo genético e o resto da população, ou seja, apenas um indivíduo cruzará com todos os outros de sua geração.

Cada indivíduo chamado de “Filho” fará quatro cruzamentos, o que resultará em quatro novos indivíduos, os quais terão sua aptidão avaliada, sendo que o mais apto substituirá seu progenitor (o parceiro de cópula do melhor indivíduo).

A Figura 14 representa o esquema utilizado no cruzamento do algoritmo genético proposto.

A forma de cruzamento segue a seguinte lógica para criação dos “Filhos”:

- No primeiro cruzamento, a primeira metade do cromossomo do melhor indivíduo é copiada para o “Filho 1” e o restante do cromossomo do “Filho 1” é composto pelos genes restantes do cromossomo do melhor indivíduo, porém na ordem em que aparecem no cromossomo indivíduo 0.

- No segundo cruzamento, a segunda metade do cromossomo do melhor indivíduo é copiada para o “Filho 2” e o restante do cromossomo do “Filho 2” é composto pelos genes restantes do cromossomo do melhor indivíduo, porém na ordem em que aparecem no cromossomo do indivíduo 0.

- No terceiro cruzamento, a primeira metade do cromossomo do indivíduo 0 é copiada para o “Filho 3” e o restante do cromossomo do “Filho 3” é composto pelos genes restantes do cromossomo do indivíduo 0, porém na ordem em que aparecem no cromossomo do melhor indivíduo.

- Finalmente, no quarto cruzamento, a segunda metade do cromossomo do indivíduo 0 é copiada para o “Filho 4” e o restante do cromossomo do “Filho 4” é composto pelos genes restantes do cromossomo do indivíduo 0, porém na ordem em que aparecem no cromossomo do melhor indivíduo.

Figura 14 - Representação de um cruzamento do algoritmo genético proposto.

Fonte: Próprio Autor.

Após todos os novos indivíduos terem sido criados, todos são avaliados e o melhor entre eles substitui o indivíduo que copulou com o melhor indivíduo obtido pelo algoritmo até então. Esta operação é realizada a cada iteração do algoritmo.

Este tipo de cruzamento oferece a vantagem de nunca gerar indivíduos inválidos, pois é garantido que todos os itens do problema estarão presentes no cromossomo do indivíduo gerado, e que a capacidade dos *bins* nunca serão excedidas. Entretanto, existe a desvantagem de haver pouca mudança no cromossomo deste indivíduo em relação aos cromossomos de seus progenitores.

3.4 APERFEIÇOAMENTO DOS INDIVÍDUOS

Cada indivíduo gerado, seja na inicialização do algoritmo, seja no cruzamento, passa por um processo de aperfeiçoamento de seu cromossomo, que visa realocar seus genes de forma a tentar utilizar o menor número de *bins*, ou seja, procura itens presentes ao longo do cromossomo de cada indivíduo sequencialmente. Isso é feito em duas direções: da direita para esquerda e da esquerda para a direita, a fim de tentar aproveitar a capacidade de cada *bin*.

A Figura 15 representa o aperfeiçoamento de um indivíduo contendo os seguintes dados referentes a um problema genérico:

- Itens a serem empacotados: 23, 33, 38, 46, 49, 57, 59, 70, 72, 73, 74, 82, 85, 92.
- Capacidade do *bin*: 150.

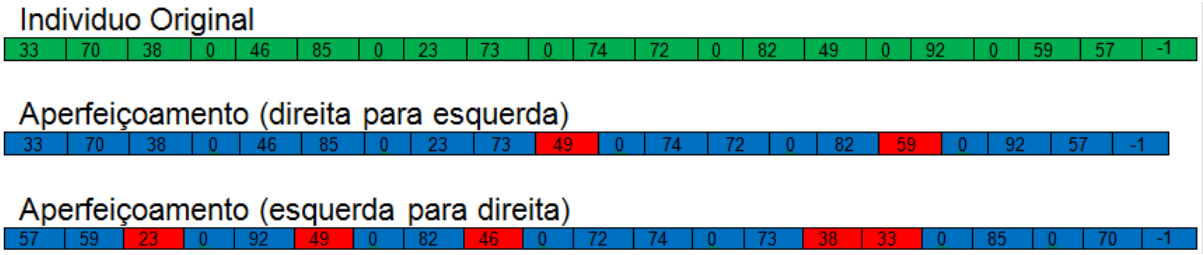

Figura 15 - Aperfeiçoamento do cromossomo de um indivíduo no algoritmo genético proposto.

Fonte: Próprio Autor.

Como se pode notar no primeiro aperfeiçoamento, o item nº 49 coube no *bin* onde estavam os itens nº 23 e nº 73 e o item nº 59 coube no *bin* onde estava o item nº 82, o que no final ocasionou uma redução de um *bin*. Porém no segundo aperfeiçoamento apesar de se conseguir aperfeiçoar os primeiros *bins*, no final não houve redução desse número.

Após finalizar o aperfeiçoamento de cada indivíduo, o melhor (o que utilizou o menor número de *bins*) é escolhido, e substitui o individuo original.

Como os itens são alocados de forma aleatória no momento da inicialização do algoritmo, esta abordagem acelera o desenvolvimento de melhores indivíduos, enquanto que no cruzamento ainda podem existir itens que caibam em um mesmo *bin* e desta forma também ajuda a minimizar o número de *bins* utilizados.

3.5 MUTAÇÃO

O método de mutação utilizado neste trabalho acontece a cada iteração do algoritmo, e seleciona aleatoriamente um gene do cromossomo de cada indivíduo para ser modificado, com exceção do melhor encontrado até então. Essa modificação ocorre da seguinte maneira: depois de selecionado o gene do indivíduo, verifica-se se este gene não é um *bin* (indicado pelo valor 0), caso não seja, será criado mais um *bin* no final do cromossomo e logo após o gene selecionado é copiado para a ultima posição do cromossomo antes do indicador de fim (-1), então os genes são rearranjados dentro do cromossomo para não haver espaço vazio. A Figura 16 que representa um empacotamento genérico, descreve passo a passo este processo.

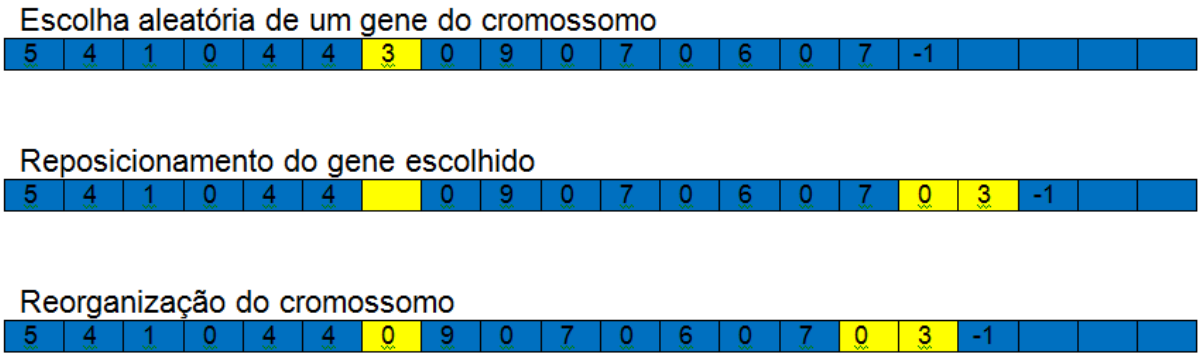

Figura 16 - Representação da mutação no algoritmo genético proposto.

Fonte: Próprio Autor.

Esse método garante que o algoritmo não caia em um ótimo local, pois apesar dos cruzamentos se estabilizarem com o passar das gerações, sempre haverá reposicionamento dos genes através da mutação e consequentemente haverá maior diversidade genética.

Com todos os operadores devidamente modificados, o algoritmo proposto foi codificado para se testar a eficiência dessa abordagem. Os resultados obtidos serão detalhados no próximo capítulo.

4 RESULTADOS

Os testes foram realizados utilizando o conjunto de entradas disponíveis no site “EURO Special Interest Group on Cutting and Packing[3]” referente ao trabalho “*A Hybrid Grouping Genetic Algorithm for Bin Packing*” de autoria de Emanuel Falkenauer (1996), que terá sua designação nas tabelas por “*HGGA*” ao passo que o algoritmo proposto será designado por “*AGM*” (Algoritmo Genético Modificado).

O conjunto de entrada compõe-se de quatro arquivos: binpacking1, binpacking2, binpacking3 e binpacking4. Cada arquivo contém vinte instâncias do problema de empacotamento unidimensional com *bins* de tamanho único, contendo itens variando de 20 a 100 unidades de “comprimento” e *bins* de 150 unidades de capacidade.

A única característica variável entre todos os arquivos é o número de itens a serem empacotados. No arquivo “binpacking1” tem-se 120, em “binpacking2” tem-se 250, em “binpacking3” tem-se 500 e por último em “binpacking4” tem-se 1000 itens. O algoritmo foi implementado em linguagem C, e os testes foram realizados em um computador com as seguintes configurações:

- Processador intel© CORE™ 2.4GHz;
- 4 GB de memória RAM;
- HD 640 GB (5400rpm);
- Windows® 7;
- IDE – BloodShed DevC++ versão 4.9.9.2.

As tabelas a seguir, apresentam os resultados encontrados para cada uma das vinte instâncias dos problemas com 120, 250, 500 e 1000 itens. As comparações são feitas entre o *HGGA* e *AGM*, em relação: ao número de *bins* encontrados, tempo gasto por instância (em segundos), média dos tempos gastos (também em segundos), número absoluto de *bins* gastos a mais pelo algoritmo proposto em relação ao *HGGA* e percentual de *bins* gastos a mais pelo algoritmo proposto em relação ao *HGGA*.

[3] Disponível em: http://paginas.fe.up.pt/~esicup/tiki-list_file_gallery.php?galleryId=1

Tabela 1 – 120 itens.

	HGGA		*AGM*			
Instância	*Bins*	Tempo	*Bins*	Tempo	Perda	Perda (%)
1	48	15.2	49	1.0	1	2,0
2	49	0.0	50	0.0	1	2,0
3	46	5.8	47	0.0	1	2,1
4	49	50.4	50	18.8	1	2,0
5	50	0.0	51	0.0	1	1,9
6	48	19.4	49	1.5	1	2,0
7	48	19.0	49	0.1	1	2,0
8	49	21.7	50	29.1	1	2,0
9	51	3668.7	51	89.8	0	0
10	46	39.5	47	5.3	1	2,1
11	52	0.0	53	1.4	1	1,8
12	49	23.7	50	0.1	1	2,0
13	48	25.7	50	0.0	2	4,0
14	49	0.0	50	0.0	1	2,0
15	50	0.0	51	1.1	1	1,9
16	48	11.1	49	1.2	1	2,0
17	52	0.0	53	0.6	1	1,8
18	52	76.1	53	7.5	1	1,8
19	49	14.3	50	0.1	1	2,0
20	50	3634.7	51	0.0	1	1,9
Média de Tempo (*HGGA*): 381			Média de Tempo (*AGM*): **8**			

No teste com 120 itens a serem empacotados, o algoritmo proposto conseguiu obter o melhor resultado na instancia número 9, encontrando o mesmo número de *bins* que o *HGGA*. O pior resultado foi obtido na instancia número 13, encontrando 2 *bins* a mais que o *HGGA.*

Em relação ao tempo de execução para o teste com 120 itens, o algoritmo proposto foi 373 segundos mais rápido que o *HGGA.*

Tabela 2 – 250 itens

	HGGA		*AGM*			
Instância	*Bins*	Tempo	*Bins*	Tempo	Perda	Perda (%)
1	99	256.7	102	4.4	3	2,9
2	100	47.4	102	15.7	2	1,9
3	102	223.8	104	156.1	2	1,9
4	100	27.0	102	169.4	2	1,9
5	101	163.5	104	4.2	3	2,8
6	101	477.6	104	32.3	3	2,8
7	102	14.5	104	11.7	2	1,9
8	104	6628.8	106	42.6	2	1,8
9	105	924.4	108	2.5	3	2,7
10	101	158.3	104	0.1	3	2,8
11	105	95.6	108	11.8	3	2,7
12	101	240.0	104	5.1	3	2,8
13	106	5996.7	108	204.4	2	1,8
14	103	6346.6	105	181.5	2	1,9
15	100	82.6	102	3.0	2	1,9
16	105	4440.1	108	213.9	3	2,7
17	97	254.5	99	103.7	2	2,0
18	100	38.5	102	12.1	2	1,9
19	100	246.8	103	3.4	3	2,9
20	102	68.0	105	4.1	3	2,8

Média de Tempo (*HGGA*): 1337 Média de Tempo (*AGM*): **59**

No teste com 250 itens a serem empacotados, o algoritmo proposto conseguiu obter o melhor resultado nas instancias número 2, 3, 4, 7, 8, 13, 14, 15, 17 e 18, encontrando dois *bins* a mais que o *HGGA*. O pior resultado foi obtido nas instancias número 1, 5, 6, 9, 10, 11, 12, 16, 19, e 20 encontrando três *bins* a mais que o *HGGA.*

Em relação ao tempo de execução para o teste com 250 itens, o algoritmo proposto foi 1258 segundos mais rápido que o *HGGA.*

Tabela 3 – 500 itens.

	HGGA		*AGM*			
Instância	*Bins*	Tempo	*Bins*	Tempo	Perda	Perda (%)
1	198	480.5	203	660.6	5	2,5
2	201	177.7	207	131.2	6	2,9
3	202	347.9	208	49.8	6	2,9
4	204	11121.2	210	266.6	6	2,9
5	206	267.6	212	205.2	6	2,8
6	206	129.7	212	84.3	6	2,8
7	207	1655.5	214	329.9	7	3,2
8	204	1834.7	211	71.9	7	3,3
9	196	501.5	201	277.7	5	2,5
10	202	92.5	207	142.5	5	2,4
11	200	106.2	206	7.1	6	2,9
12	200	152.3	206	98.2	6	2,9
13	199	1019.3	205	174.8	6	2,9
14	196	135.5	202	36.6	6	3,0
15	204	951.7	209	969.1	5	2,4
16	201	375.2	206	942.3	5	2,4
17	202	162.6	207	96.5	5	2,4
18	198	336.8	204	14.2	6	2,9
19	202	143.9	207	598.5	5	2,4
20	196	306.8	202	26.6	6	3,0
Média de Tempo (*HGGA*): 1015			Média de Tempo (*AGM*): **259**			

No teste com 500 itens a serem empacotados, o algoritmo proposto conseguiu obter o melhor resultado nas instancias número 1, 9, 10, 15, 16, 17 e 19 encontrando cinco *bins* a mais que o *HGGA*. O pior resultado foi obtido nas instancias número 7 e 8, encontrando sete *bins* a mais que o *HGGA.*

Em relação ao tempo de execução para o teste com 500 itens, o algoritmo proposto foi 756 segundos mais rápido que o *HGGA.*

Tabela 4 – 1000 itens

HGGA			*AGM*			
Instância	*Bins*	Tempo	*Bins*	Tempo	Perda	Perda (%)
1	399	2924.7	409	5318.5	10	2,4
2	406	4040.2	418	2104.7	12	2,9
3	411	6262.1	424	458.1	13	3,1
4	411	32714.3	424	1097.1	13	3,1
5	397	11862.0	408	5509.9	11	2,7
6	399	3774.3	410	4267.0	11	2,7
7	395	3033.2	406	7578.7	11	2,7
8	404	9878.8	415	6686.5	11	2,7
9	399	5585.2	411	1252.7	12	2,9
10	397	8126.2	409	2737.6	12	2,9
11	400	3359.1	412	1012.7	12	2,9
12	401	6782.3	413	7199.5	12	2,9
13	393	2537.4	404	2738.8	11	2,7
14	396	11828.8	407	7530.7	11	2,7
15	394	5838.1	406	3386.7	12	3,0
16	402	12610.8	415	523.9	13	3,1
17	404	2740.8	415	11070.0	11	2,7
18	404	2379.4	416	2482.0	12	2,9
19	399	1329.7	411	367.0	12	2,9
20	400	3564.2	412	7797.7	12	2,9
Média de Tempo (*HGGA*): 7059			Média de Tempo (*AGM*): **4056**			

No teste com 1000 itens a serem empacotados, o algoritmo proposto conseguiu obter o melhor resultado na instancia número 1, encontrando dez *bins* a mais que o *HGGA*. O pior resultado foi obtido nas instancias número 3, 4 e 16, encontrando treze *bins* a mais que o *HGGA.*

Em relação ao tempo de execução para o teste com 1000 itens, o algoritmo proposto foi 3003 segundos mais rápido que o *HGGA.*

5 CONCLUSÃO

Como foi apresentado pelas tabelas anteriores, o algoritmo proposto não conseguiu obter melhores resultados em relação ao número de *bins* encontrados por Falkenauer (1996), ficando em média 2,48% acima do "*HGGA*". Porém, o "*AGM*" conseguiu em geral, ser mais rápido em sua execução.

Uma das possíveis causas do desempenho inferior do "*AGM*" em relação ao "*HGGA*" quando se trata do número de *bins* encontrados, se deve ao fato de que o material genético gerado pelo primeiro, é muito homogêneo, uma vez que apenas um indivíduo realiza o cruzamento com todos os outros da população, portanto, os indivíduos da próxima geração são todos muito parecidos (pois possuem o mesmo "pai"). Em compensação, a utilização da função que tenta aperfeiçoar os cromossomos dos indivíduos, resultou em uma rápida convergência para os melhores resultados, o que fez com que o algoritmo proposto fosse relativamente mais rápido em sua execução.

Levando-se em consideração a comparação percentual entre os resultados obtidos pelo "*AGM"* em relação ao "*HGGA"* e a média dos tempos de execução entre os dois algoritmos, pode-se considerar que o algoritmo proposto encontrou boas soluções e que pode ser implementado em situações práticas com garantia de obtenção de bons resultados.

6 TRABALHOS FUTUROS

Uma sugestão para um possível trabalho futuro referente ao tema estudado seria o desenvolvimento de um algoritmo híbrido entre o *HGGA* e o algoritmo proposto neste trabalho, uma vez que o primeiro obteve melhores resultados em relação à utilização do menor número de *bins* para acomodar os itens, e o segundo obteve um melhor tempo de execução na média geral dos testes realizados. Desta forma, um possível algoritmo híbrido entre os dois comparados neste trabalho, poderia encontrar resultados que combinasse o melhor dos dois algoritmos e conseguisse melhorar todos os aspectos relevantes ao problema estudado.

REFERÊNCIAS

ALECRIM, Emerson. **Características e Funcionamento dos HDs (Discos Rígidos)**. Disponível em: http://www.infowester.com/hd.php. Acessado em: 17/09/2012.

ASCENCIO, Ana Fernanda Gomes; ARAÚJO, Graziela Santos de. **Estrutura de Dados: Algoritmos, Análise da Complexidade e Implementações em Java e C/C++.** Editora Pearson Prentice Hall, 2010.

CAVALI, Roberto. **Problemas de Corte e Empacotamento na Indústria de Móveis: Um Estudo de Caso.** Dissertação de Mestrado – Instituo de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista. 2004. Disponível em: http://www.athena.biblioteca.unesp.br/exlibris/bd/brp/33004153071P0/2004/cavali_r_me_sjrp.pdf. Acessado em: 02/08/2012.

CORMEN, Tomas H.; LEISERSON, Charles E.; RIVEST, RONALD L.; STEIN, Clifford. **Algoritmos Teoria e Prática**. Rio de Janeiro. Editora Elsevier, 6ª Tiragem, 2002.

DICKHOFF, Harald. **A Typology of Cutting and Packing Problems**. European Jornal of Operational Research, 1990.

DIJKSTRA, E. W. **A Short Introduction to the Art of Programing.** Technological University Endhoven, 1971.

FALKENAUER, Emanuel. **A Hybrid Grouping Genetic Algorithm for Bin Packing**. Journal of Heuristics. 1996. Disponível em: http://reference.kfupm.edu.sa/content/h/y/a_hybrid_grouping_genetic_algorithm_for_61460.pdf. Acessado em: 19/10/2012.

FARIA, Anderson Oliveira. **Otimização do Problema de Corte e Empacotamento Unidimensional Utilizando Algoritmo Genético**. Trabalho de Conclusão de Curso (Gradução) – Departamento de Ciência da Computação. Universidade Federal de Lavras - MG. 2006. Disponível em: http://www.bcc.ufla.br/monografias/2005/Otimizacao_do_problema_de_corte_e_empacotamento_unidimensional_utilizando_algoritmo_genetico.pdf. Acessado em: 03/08/2012.

FONSECA FILHO, Cléuzio. **História da Computação: O Caminho do Pensamento e da Tecnologia**. Porto Alegre. Editora EDIPUCRS, 2007.

GRONER, L. **NP-Completude**. Fundação de Assistência e Educação – FAESA. Faculdades Integradas Espírito-Santeses. Vitória. Disponível em: http://www.loiane.com/2009/08/problemas-p-versus-np/. Acessado em: 26/06/2012.

HARDESTY, Larry. **Explained: P vs. NP**. The most notorious problem in theoretical computer science remains open, but the attempts to solve it have led to profound

insights. Disponível em: http://web.mit.edu/newsoffice/2009/explainer-pnp.html. Acessado em 16 de maio de 2012.

HEIS, Adriano. **Um Algoritmo Evolutivo para o Problema de Corte de Estoque Unidimensional Inteiro**. Dissertação de Mestrado. Universidade Estadual de Maringá. 2007. Disponível em: www.din.uem.br/arquivos/pos-graduacao/mestrado-em-ciencia-da-computacao/dissertacoes/Um Algoritmo Evolutivo para o Problema de Corte de Estoque Unidimensional Inteiro (Adriano Heis).pdf. Acessado em: 07/07/2012.

HOLLAND, John. **Adaptation in Natural and Artificial Systems**. MIT Press Cambrige, MA, USA. 1992.

JOHNSON, David. S. **The NP-Completeness column: an ongoing guide.** Bell Laboratories, Murray Hill, New Jersey 1986. Disponível em: http://www2.research.att.com/~dsj/columns/col18.pdf. Acessado em: 12/07/2012.

LINDEN, Ricardo. **Algoritmos Genéticos. Uma Importante Ferramenta da Inteligência Computacional**. 2ª edição. Rio de Janeiro. Editora BRASPORT, 2008.

MIYAZAWA, Flávio Keidi. **Otimização Combinatória**. Disponível em www.ic.unicamp.br/~fkm/problems/combopt.html. Acessado em 16 de maio de 2012.

MONTEVECHI, **José Arnaldo Barra. Pesquisa Operacional (Programação Linear)**. Escola Federal de Engenharia de Itajubá. 2000. Disponível em: http://www.inf.ufpr.br/ess07/Meus_Programas/PO/Textos/apostilaepr05.pdf.pdf. Acessado em: 13/06/2012.

MORABITO, Reinaldo. **Modelos de Otimização para o Problema de Corte nas Indústrias de Papel e Papelão e de Móveis**. Departamento de Engenharia de Produção. Universidade Federal de São Carlos. Disponível em: http://www.dep.ufscar.br/docentes/morabito/g&p94.pdf. Acessado em: 29/06/2012.

MORAIS, José Luiz Machado. **Problema do caixeiro viajante aplicado ao roteamento de veículos numa malha viária**. Trabalho de Conclusão de Curso (Graduação) – Instituto de Ciência e Tecnologia. Universidade Federal de São Paulo. Disponível em: http://www.luismeira.com.br/orientacoes/TCC_JoseLuiz.pdf. Acessado em: 22/06/2012.

NORONHA, Tiago Ferreira. **Um Algoritmo Memético de Grupamento para o Problema de Bin Packing 1-D**. Revista eletrônica de iniciação científica. 2001. Disponível em: http://www.sbc.org.br/reic/edi. Acessado em: 03/07/2012.

OLIVEIRA, Sara. **Reprodução Sexuada e Variabilidade Genética**. Disponível em: http://e-porteflio.blogspot.com.br/2008/11/reproduo-sexuada-e-variabilidade.html. Acessado em: 02/11/2012.

ROCHA, Guaracy Tadeu. **Transmissão da Vida, Ética e Manipulação Gênica**. 2012. Disponível em:

http://www2.ibb.unesp.br/nadi/Museu5_transmissao/Museu5_transmissao.htm. Acessado em: 14/09/2012.

SOARES, Gustavo Luíz. **Algoritmos Genéticos: Estudo, Novas Técnicas e Aplicações.** Escola de Engenharia da Universidade de Minas Gerais. 1997. Disponível em: http://www.cpdee.ufmg.br/~joao/TesesOrientadas/VAS1997_1.pdf. Acessado em: 12/08/2012.

TOSCANI, Laira Vieira; VELOSO, Paulo. A. S. **Complexidade de Algoritmos**. Instituto de Informática da Universidade Federal do Rio Grande Do Sul, 1ª edição. Porto Alegre. Editora Sagra Luzzatto, 2002.

VINK, Marco. **Solving Combinatorial Problems Using Evolutionary Algorithms**. 1997. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.51.4300. Acessado em: 22/09/2012.

WÄSCHER, Gerhard; HAUBNER, Heike; SCHUMANN, Holger. **An Improved Typology of Cutting and Packing Problems**. European Jornal of Operational Research, 2007.

XAVIER, Eduardo. C. **Algoritmos para Problemas de Empacotamento**. Tese de Doutorado. Instituto de Computação. Universidade Estadual de Campinas, 2006. Disponível em: http://www.ic.unicamp.br/~eduardo/interno/Publications_files/tese.pdf. Acessado em: 20/07/2012.

ZIVIANI, Nivio. **Projeto de Algoritmos: com Implementações em Pascal e C.** 3ª edição. São Paulo. Editora Cengage Learning, 2010.