Aplicação de Algoritmos Genéticos para Solução do Problema da Caixa Preta

Flávia Gonçalves Fernandes[1], Marcos Napoleão Rabelo[1], Sérgio Francisco da Silva[1]

[1]Universidade Federal de Goiás (UFG) – Catalão – GO – Brasil

flavia.fernandes92@gmail.com, rabelomn@gmail.com, sergio.f.silva@gmail.com

Abstract. *Much of the cost in the software life program concentrates on maintenance of them. To mitigate this situation, software testing is used for an application error purpose, avoiding end-user dissatisfaction, disruption and expense to the company. In this perspective, the black box problem is a software test to check the output of the data using the inputs of various types. Such entries are not chosen according to the structure of the program. Thus, this work has an implementation of a genetic algorithm to solve the problem of the Black box and presents as results a study of a simplified method of some of the parameters in the performance of the algorithm implemented.*

Resumo. *Grande parte dos custos no ciclo de vida de software se concentram na manutenção dos mesmos. Para amenizar esta situação, os testes de software são utilizados com a finalidade de reduzir os erros de aplicação, evitando o descontentamento do usuário final, transtornos e gastos à empresa. Nessa perspectiva, o problema da caixa-preta é um teste de software para verificar a saída dos dados usando entradas de vários tipos. Tais entradas não são escolhidas conforme a estrutura do programa. Desse modo, este trabalho tem como objetivo a implementação de um Algoritmo Genético para a resolução do Problema da Caixa Preta e apresenta como resultados a realização de um estudo simplificado dos efeitos de alguns dos parâmetros no desempenho do algoritmo implementado.*

1. Introdução

O termo caixa preta vem originalmente das telecomunicações militares e dos equipamentos inimigos que não podiam ser abertos devido a possibilidade de conter explosivos. Um sistema formado por módulos que cumpram as características de caixa preta simplifica a compreensão do funcionamento e permitem dar uma visão mais clara do conjunto. Esse sistema é ainda mais robusto, fácil de manter; e, em caso de alguma falha, este poderá ser analisado e abordado de maneira mais ágil [Rene, 2004].

A metodologia de abordagem de caixa preta utilizada para analisar um sistema faz uso apenas da análise da relação entre o estímulo de entrada e a resposta de saída. A causalidade não é assumida, mas é uma hipótese simplificadora. Idealmente, a descrição

matemática dessas relações permitem conclusões sobre a natureza das relações dos sinais com o sistema [Rene, 2004].

Em programação modular, onde um programa (ou um algoritmo) é dividido em módulos, na fase de projeto, procura-se desenvolver cada módulo como uma caixa preta dentro do sistema global que o sistema pretende desempenhar. Desta maneira consegue-se uma independência entre os módulos e facilita-se sua implementação separada por uma equipe de trabalho onde cada membro encarrega-se de implementar uma parte (um módulo) do programa global; assim o implementador de um módulo concreto deverá conhecer como é a comunicação dos outros módulos (a interface), mas não necessitará conhecer como trabalham esses módulos internamente [Linden, 2008].

Nessa linha de raciocínio, o objetivo principal deste trabalho é implementar um algoritmo genético para a resolução do problema da caixa preta e realizar um estudo simplificado dos efeitos de alguns parâmetros no desempenho do algoritmo desenvolvido.

Na seção 2, é apresentada a conceituação teórica sobre o problema da caixa preta e seus respectivos fundamentos. A seção 3 aborda a metodologia utilizada no desenvolvimento do trabalho. Na seção 4, os resultados obtidos são expostos e, por fim, a seção 5 mostra as conclusões e trabalhos futuros da pesquisa.

2. Fundamentação Teórica

Um algoritmo genético (AG) é uma técnica de busca utilizada na ciência da computação para achar soluções aproximadas em problemas de otimização e busca, fundamentado principalmente pelo americano John Henry Holland. Algoritmos genéticos são uma classe particular de algoritmos evolutivos que usam técnicas inspiradas pela biologia evolutiva como hereditariedade, mutação, seleção natural e recombinação (ou *crossing over*) [Goldberg, 2009].

Algoritmos genéticos são implementados como uma simulação de computador em que uma população de representações abstratas de solução é selecionada em busca de soluções melhores. A evolução geralmente se inicia a partir de um conjunto de soluções criado aleatoriamente e é realizada por meio de gerações. A cada geração, a adaptação de cada solução na população é avaliada, alguns indivíduos são selecionados para a próxima geração, e recombinados ou mutados para formar uma nova população. A nova população então é utilizada como entrada para a próxima iteração do algoritmo [Norvig; Russel, 2009].

A função-objetivo é o objeto da otimização. Pode ser um problema de otimização, um conjunto de teste para identificar os indivíduos mais aptos, ou mesmo uma "caixa preta" onde sabemos apenas o formato das entradas e nos retorna um valor que queremos otimizar. A grande vantagem dos algoritmos genéticos está no fato de não ser necessário saber como funciona esta função objetivo, apenas tê-la disponível para ser aplicada aos indivíduos e comparar os resultados [Koza, 2002].

O indivíduo é meramente um portador do seu código genético. O código genético é uma representação do espaço de busca do problema a ser resolvido, em geral na forma de sequências de bits. Por exemplo, para otimizações em problemas cujos valores de entrada são inteiros positivos de valor menor que 255 podemos usar 8 bits, com a representação binária normal, ou ainda uma forma de código *gray*. Problemas com múltiplas entradas podem combinar as entradas em uma única sequência de bits, ou

trabalhar com mais de um "cromossomo", cada um representando uma das entradas. O código genético deve ser uma representação capaz de representar todo o conjunto dos valores no espaço de busca, e precisa ter tamanho finito [Collares, 2017].

A seleção também é outra parte chave do algoritmo. Em geral, usa-se o algoritmo de seleção por "roleta", onde os indivíduos são ordenados de acordo com a função-objetivo e lhes são atribuídas probabilidades decrescentes de serem escolhidos - probabilidades essas proporcionais à razão entre a adequação do indivíduo e a soma das adequações de todos os indivíduos da população. A escolha é feita então aleatoriamente de acordo com essas probabilidades. Dessa forma conseguimos escolher como pais os mais bem adaptados, sem deixar de lado a diversidade dos menos adaptados. Outras formas de seleção podem, ainda, ser aplicadas dependendo do problema a ser tratado. Como exemplos pode-se citar a seleção por "torneio" (onde são selecionados diversos pequenos subconjuntos da população, sendo selecionado o indivíduo de maior adequação de cada um desses grupos), a seleção por "classificação" ou "ranking" (semelhante à seleção por "roleta", com a diferença de que a probabilidade de seleção é relacionada à sua posição na ordenação dos indivíduos da população e não à sua adequação em si) e a seleção por "truncamento" (onde são selecionados os N melhores indivíduos da população, descartando-se os outros) [Goldberg, 2009].

A reprodução, tradicionalmente, é dividida em três etapas: acasalamento, recombinação e mutação. O acasalamento é a escolha de dois indivíduos para se reproduzirem (geralmente gerando dois descendentes para manter o tamanho populacional). A recombinação, ou crossing-over é um processo que imita o processo biológico homônimo na reprodução sexuada: os descendentes recebem em seu código genético parte do código genético do pai e parte do código da mãe. Esta recombinação garante que os melhores indivíduos sejam capazes de trocar entre si as informações que os levam a ser mais aptos a sobreviver, e assim gerar descendentes ainda mais aptos. Por último vem as mutações, que são feitas com probabilidade a mais baixa possível, e tem como objetivo permitir maior variabilidade genética na população, impedindo que a busca fique estagnada em um mínimo local [Linden, 2008].

Logo, os algoritmos genéticos são algoritmos de otimização global, baseados nos mecanismos de seleção natural e da genética. Eles empregam uma estratégia de busca paralela e estruturada, mas probabilística, que é voltada em direção ao reforço da busca de pontos de "alta aptidão".

3. Metodologia

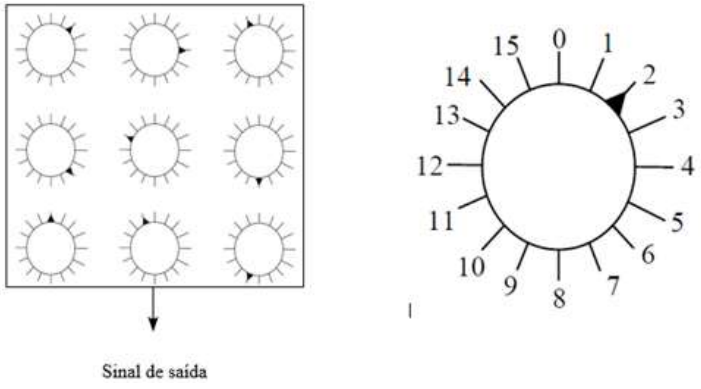
Para o desenvolvimento do sistema que busca a solução para o Problema da Caixa Preta, foi implementado um algoritmo genético utilizando linguagem de programação Java e paradigma de programação orientada a objetos. A ferramenta de desenvolvimento adotada foi o *software* Eclipse [Eclipse, 2017].

Desse modo, para a realização desta implementação, primeiramente, foram pesquisados os conceitos sobre o problema da caixa preta e de algoritmos genéticos. Posteriormente, foram pesquisados algoritmos e códigos-fonte já existentes em linguagem de programação Java, com a finalidade de auxiliar no desenvolvimento do problema proposto, devido à sua complexidade.

A partir dos conhecimentos adquiridos e das informações coletadas em [Carvalho; Lacerda, 2017], [Collares, 2017], [Linden, 2008], [Maia, 2017], [Souza,

2017] e [Eclipse, 2017], foi possível a implementação deste Algoritmo Genético para solução do Problema da Caixa Preta.

Sabendo-se que cada botão da caixa preta apresentada na Figura 1 pode ser colocado em dezesseis posições distintas, e que se deseja maximizar o sinal de saída, deseja-se encontrar a melhor combinação de posições.



(a) **(b)**

Figura 1. (a) Caixa Preta. (b) Cada botão da Caixa Preta pode ser colocado em uma das 16 posições disponíveis. Cada combinação de posições gera um sinal de saída diferente.

Sabendo-se que cada botão da Caixa Preta pode ser colocado em uma de dezesseis posições (posições de zero a quinze), cada posição pode ser representada por um número binário de quatro bits. A posição 2, por exemplo, apresentada na parte (b) da Figura 1, é representada pelo número 0010. A Tabela 1 apresenta as representações para cada uma das dezesseis posições possíveis para cada um dos botões da Caixa Preta. Cada indivíduo (cromossomo) representará uma combinação de posições para os botões e, portanto, possuirá trinta e seis bits (quatro bits para cada um dos nove botões), totalizando 236 (aproximadamente 68,72 x 109) soluções possíveis. Considerando-se essa codificação, um indivíduo representando a combinação de posições apresentada na parte (a) da Figura 1 apresenta a seguinte estrutura: 001001001111011011011000000011111001 (2, 4, 15, 6, 13, 8, 0, 15, 9).

Tabela 1. Representações para cada uma das posições possíveis para os botões da caixa preta.

Posição	Representação	Posição	Representação
0	0000	8	1000
1	0001	9	1001
2	0010	10	1010
3	0011	11	1011
4	0100	12	1100
5	0101	13	1101
6	0110	14	1110
7	0111	15	1111

No problema real, o mapeamento entre as 236 posições é desconhecido: só se tem acesso ao valor de saída após a aplicação de uma configuração de posições para os botões (motivo para a denominação Caixa Preta).

Assim, a solução ótima, determinada por meio da análise da função utilizada para simulação (no problema real, a solução ótima é desconhecida), cujo valor correspondente de saída é vinte e sete, é 11111011100110111111011111100101111.

Dessa maneira, o Algoritmo Genético implementado apresenta as seguintes características:

- Seleção por roleta ou por torneio de 2 indivíduos (opção do usuário).
- Mutação por escolha aleatória do bit ou bit a bit (escolha do usuário).
- Possibilidade de configurar as probabilidades de cruzamento/crossover e mutação, o número de indivíduos da população e o número de gerações.

4. Resultados

Após a implementação deste Algoritmo Genético com a finalidade de solucionar o Problema da Caixa Preta, foram realizados alguns testes de execução do mesmo.

Para os testes a serem realizados, uma execução foi considerada como um sucesso quando conseguiu encontrar a solução ótima (sinal de saída = 27). Para cada configuração de parâmetros, foram realizadas 100 (cem) execuções do algoritmo e apresentados as seguintes informações:

- Número de sucessos (quantidade de vezes que o algoritmo genético encontrou o valor de saída 27);
- Maior valor de fitness/avaliação (melhor caso quando não encontrar o valor ótimo);
- Menor valor de fitness/avaliação (pior caso quando não encontrar o valor ótimo);
- Valor médio de fitness/avaliação (média dos melhores resultados das 100 execuções).

4.1. Primeiro Teste

No Primeiro Teste, foi realizada a avaliação do tipo de cruzamento (*crossover*), utilizando o crossover com um ponto de corte, isto é, para cada par de indivíduos selecionados para o crossover, foi sorteado um ponto de corte. Além disso, foram adotadas as seguintes configurações como parâmetros: com elitismo (100%); seleção por roleta; mutação bit a bit; taxa de crossover: 80%; taxa de mutação: 2,5% de chances cada bit ser selecionado; número de indivíduos da população: 30; número de gerações: 50.

Após a execução da 50ª geração do Primeiro Teste com as configurações especificadas acima, foram encontrados os seguintes resultados: número de sucessos =

24; maior valor de fitness/avaliação = 27; menor valor de fitness/avaliação = 26; valor médio de fitness/avaliação = 26.8.

4.2. Segundo Teste

No Segundo Teste, foi realizada a avaliação do efeito do tipo de seleção, executando o algoritmo genético e utilizando, primeiramente, seleção por Roleta. E, posteriormente, seleção por Torneio de 2. Além disso, foram adotadas as seguintes configurações como parâmetros: com elitismo (100%); crossover com um ponto de corte; mutação bit a bit; taxa de crossover: 80%; taxa de mutação: 2,5% de chances cada bit ser selecionado; número de indivíduos da população: 30; número de gerações: 50.

Após a execução da 50ª geração do Segundo Teste utilizando o método de seleção por Roleta com as configurações especificadas acima, foram encontrados os seguintes resultados: número de sucessos = 30; maior valor de fitness/avaliação = 27; menor valor de fitness/avaliação = 27; valor médio de fitness/avaliação = 27.

Após a execução da 50ª geração do Segundo Teste utilizando o método de seleção por Torneio de 2 com as configurações especificadas acima, foram encontrados os seguintes resultados: número de sucessos = 0; maior valor de fitness/avaliação = 26; menor valor de fitness/avaliação = 25; valor médio de fitness/avaliação = 25.

Logo, pode-se observar que a seleção por roleta é mais eficiente do que a seleção por torneio de 2. Essa conclusão foi obtida devido à otimização do algoritmo genético para o problema da caixa preta apresentar melhores resultados por meio da seleção por roleta do que por meio da seleção por torneio de 2. Para isso, considerou-se que só o parâmetro de seleção foi alterado para a execução do sistema e que, a cada execução, o algoritmo genético apresenta resultados diferentes em virtude dos vários parâmetros que são gerados aleatoriamente (população inicial, ponto de corte de crossover, bit de mutação).

4.3. Terceiro Teste

No Terceiro Teste, foi realizada a avaliação do efeito do tipo de mutação, executando o algoritmo genético e utilizando, primeiramente, mutação bit a bit. E, posteriormente, mutação por escolha aleatória do bit. Além disso, foram adotadas as seguintes configurações como parâmetros: com elitismo (100%); crossover com um ponto de corte; seleção por roleta; taxa de crossover: 80%; taxa de mutação: 2,5% de chances cada bit ser selecionado; número de indivíduos da população: 30; número de gerações: 50.

Após a execução da 50ª geração do Terceiro Teste utilizando o método de mutação bit a bit com as configurações especificadas acima, foram encontrados os seguintes resultados: número de sucessos = 0; maior valor de fitness/avaliação = 26; menor valor de fitness/avaliação = 25; valor médio de fitness/avaliação = 25.43.

Após a execução da 50ª geração do Terceiro Teste utilizando o método de mutação por escolha aleatória do bit com as configurações especificadas acima, foram encontrados os seguintes resultados: número de sucessos = 30; maior valor de

fitness/avaliação = 27; menor valor de fitness/avaliação = 27; valor médio de fitness/avaliação = 27.

Logo, pode-se observar que a mutação por escolha aleatória do bit é mais eficiente do que a mutação bit a bit. Essa conclusão foi obtida devido à otimização do algoritmo genético para o problema da caixa preta apresentar melhores resultados por meio da mutação aleatória do bit do que por meio da mutação bit a bit. Para isso, considerou-se que só o parâmetro de mutação foi alterado para a execução do sistema e que, a cada execução, o algoritmo genético apresenta resultados diferentes em virtude dos vários parâmetros que são gerados aleatoriamente (população inicial, ponto de corte de crossover, bit de mutação).

4.4. Quarto Teste

No Quarto Teste, foi realizada a avaliação do efeito da probabilidade do cruzamento, executando o algoritmo genético e utilizando taxa de crossover iguais a 20%, 50% e 80%. Além disso, foram adotadas as seguintes configurações como parâmetros: com elitismo (100%); crossover com um ponto de corte; seleção por roleta; mutação por escolha aleatória do bit; taxa de mutação: 2,5% de chances cada bit ser selecionado; número de indivíduos da população: 30; número de gerações: 50.

Após a execução da 50ª geração do Quarto Teste utilizando a taxa de crossover igual a 20% com as configurações especificadas acima, foram encontrados os seguintes resultados: número de sucessos = 0; maior valor de fitness/avaliação = 21; menor valor de fitness/avaliação = 20; valor médio de fitness/avaliação = 20.57.

Após a execução da 50ª geração do Quarto Teste utilizando a taxa de crossover igual a 50% com as configurações especificadas acima, foram encontrados os seguintes resultados: número de sucessos = 22; maior valor de fitness/avaliação = 27; menor valor de fitness/avaliação = 25; valor médio de fitness/avaliação = 26.2.

Após a execução da 50ª geração do Quarto Teste utilizando a taxa de crossover igual a 80% com as configurações especificadas acima, foram encontrados os seguintes resultados: número de sucessos = 30; maior valor de fitness/avaliação = 27; menor valor de fitness/avaliação = 27; valor médio de fitness/avaliação = 27.

Logo, pode-se observar que quanto maior for a taxa de crossover, melhor a solução encontrada. Essa conclusão foi obtida devido à otimização do algoritmo genético para o problema da caixa preta apresentar melhores resultados por meio da taxa de crossover igual a 80% do que em relação aos testes utilizando 20% e 50%. Para isso, considerou-se que apenas o parâmetro de taxa de crossover foi alterado para a execução do sistema e que, a cada execução, o algoritmo genético apresenta resultados diferentes em virtude dos vários parâmetros que são gerados aleatoriamente (população inicial, ponto de corte de crossover, bit de mutação).

5. Conclusões e Trabalhos Futuros

Portanto, percebe-se que é necessária uma população mínima de 30 indivíduos e a número de gerações igual a 50 para o algoritmo genético apresentar melhor otimização.

Valores inferiores a estes estipulados interferem no algoritmo genético, interferindo de maneira negativa nos resultados desejados para otimização do sistema.

Em virtude do que foi pesquisado e realizado, verifica-se que os algoritmos genéticos são muito úteis como, por exemplo: controle de sistemas dinâmicos; indução e otimização de bases de regras; encontrar novas topologias conexionistas; engenharia de sistemas neurais artificiais; modelagem de estruturas neurais biológicas e simulação de modelos biológicos. Estas aplicações são de suma importância para o progresso de projetos que envolvem inteligência artificial.

Como trabalhos futuros, espera-se aplicar a implementação de algoritmos genéticos para solucionar outros problemas dessa natureza com a finalidade de almejar a otimização do sistema. Além disso, observa-se que este trabalho pode promover um aprendizado do conteúdo sobre algoritmos genéticos de uma maneira mais dinâmica, pois propõe uma problemática prática e real, o que torna o seu desenvolvimento mais atrativo e agradável, e fixando de forma mais eficaz os conceitos teóricos sobre o tema.

Referências

Carvalho, André Carlos P. L. F.; Lacerda, Estéfane G. M.. Introdução aos algoritmos genéticos. Disponível em: <http://www.leca.ufrn.br/~estefane/metaheuristicas/ag.pdf>. Acesso em: 01 nov. 2017.

Collares, Paulo. Algoritmo genético clássico em java. Disponível em: <http://www.paulocollares.com.br/algoritimo-genetico-classico-em-java-hello-world/>. Acesso em: 01. nov. 2017.

___. Eclipse. Disponível em: <www.eclipse.org/>. Acesso em: 01 nov. 2017.

Goldberg, David E.. Genetic Algorithms in Search, Optimization, and Machine Learning. EUA: Addison-Wesley, 2009.

Koza, J.R.. Genetic Programming: On the Programming of Computers by Means of Natural Selection. [S.l.]: MIT Press, 2002.

Linden, Ricardo. Algoritmos Genéticos - uma importante ferramenta da inteligência computacional - 2ª Edição. BR: Brasport, 2008.

Maia, Renato Dourado. Trabalho prático da disciplina de algoritmos genéticos. Disponível em: <http://www.cpdee.ufmg.br/~rdmaia/2011/01/GENETICOS/TrabalhoPratico.pdf>. Acesso em: 01 nov. 2017.

Norvig, Peter; Russel, Stuart. Artificial Intelligence: A Modern Aproach. Upper Saddle River, NJ, EUA: Prentice Hall, 2009.

Rene, Thom. Mathematical Models of Morphogenesis. Chichester: Ellis Horwood, 2004.

Souza, Marcone Jamilson Freitas. Inteligência computacional para otimização. Disponível em: <http://www.decom.ufop.br/prof/marcone/Disciplinas/InteligenciaComputacional/InteligenciaComputacional.htm>. Acesso em: 01. nov. 2017.