Otimização do COCOMO Básico utilizando Algoritmo Genético para estimativa de esforço no desenvolvimento de software.

Arielson Altino de Souza[1], Marco Antônio Pereira Araújo[1], Márcia Cristina Valle Zanetti[1]

[1]Instituto Federal do Sudeste de Minas
Campus Juiz de Fora -- MG – Brasil

arielsonalt@gmail.com,{marco.araujo, marcia.zanetti}@ifsudestemg.edu.br

Abstract. *Software development has greatly increased and being able to estimate costs has become a difficult task. Metrics have been widely used and proposed to estimate the effort in software development. The Basic COCOMO metric has been used as a research base in order to optimize its parameters. Artificial intelligence techniques are used by authors to seek better estimates, among them the Genetic Algorithm (GA) produces good results presented in the literature. This research proposes a Genetic Algorithm model to optimize Basic COCOMO parameters. The result shows that the proposed algorithm model presents in terms of Mean Absolute Error (MAE) 93.517% improvement compared to Basic COCOMO, 41.639% improvement compared to another proposed GA model, 2.515% improvement compared to Particle Swarm Optimization (PSO) and 1.594% improvement compared to the proposed Nature-Inspired Algorithm (NIA) model.*

Resumo. O desenvolvimento de software tem aumentado em larga escala e ser capaz de estimar os custos se tornou uma tarefa difícil. Métricas têm sido amplamente utilizadas e propostas para estimar o esforço no desenvolvimento de software. A métrica COCOMO Básico tem sido utilizada como base de pesquisas com o objetivo de otimizar seus parâmetros. Técnicas de inteligência artificial são usadas por autores para buscar melhores estimativas, entre elas o Algoritmo Genético (AG) produz bons resultados apresentados na literatura. Esta pesquisa propõe um modelo de Algoritmo Genético para otimizar os parâmetros do COCOMO Básico. O resultado mostra que o modelo proposto de algoritmo apresenta em termos de *Mean Absolute Error* (MAE) melhoria de 93,517% comparado ao COCOMO Básico, 41,639% de melhoria comparado a outro modelo de AG proposto em 2006, 2,515% de melhoria comparado ao modelo proposto de *Particle Swarm Optimization* (PSO) e 1,594% de melhoria comparado ao modelo proposto de *Nature-Inspired Algorith*m (NIA).

Palavras-chave: engenharia de software, algoritmo genético, COCOMO básico, estimativa de esforço.

1. Introdução

No início dos anos 2000, mais precisamente no ano de 2001 o manifesto sobre *search-based software engineering* foi apresentado por Mark Harman e Bryan F. Jones [Harman e Jones 2001]. A natureza complexa dos problemas contidos em engenharia de *software* é ideal para utilizar os algoritmos de busca. Problema de busca baseado em gerenciamento de projetos de *softwares* provê trabalhos em muitas áreas de engenharia de *software* como

Gerenciamento de Tempo, custo, qualidade, recursos humanos e risco [Ferrucci et al. 2014]. A resolução não computacional de tais problemas não é recomendada, pois as variáveis envolvidas estão interligadas de maneira que uma afeta diretamente ou indiretamente outras.

As questões abordadas pela engenharia de *software* se assemelham a questões de outras engenharias como por exemplo o fato de algumas soluções requererem algo próximo do ótimo ou com algum tipo de tolerância. Meta-heurísticas são indicadas para problemas considerados de complexidade alta. Tais algoritmos como *simulated annealing*, algoritmo genético e busca tabu têm sido usados em vários trabalhos hoje encontrados na literatura. A literatura dispõe de abordagens utilizando o algoritmo genético para área de teste, estimativa de custo entre outros [Harman e Jones 2001].

O desenvolvimento de *software* tem aumentado em larga escala e ser capaz de estimar os custos se tornou uma tarefa difícil [Sheta 2006]. Modelos para estimativa do esforço no desenvolvimento são amplamente utilizadas para otimizar o gerenciamento de recursos durante todo o desenvolvimento de *software* [Chirra et al. 2019]. A importância de estimar o esforço no desenvolvimento de software está ligada ao fato de tornar possível a utilização dos recursos de maneira controlada e auxiliar na gerência dos projetos de *software* [Chirra et al. 2019].

Durante as últimas décadas, pesquisas apresentaram modelos algorítmicos com o objetivo de obter acurácia na Estimativa de Esforço de *Software* (EES) [Dhiman e Diwaker 2013]. Os autores das pesquisas abordam a utilização de vários métodos como *Constructive Cost Model* (COCOMO) [Singh e Misra 2012, Sheta 2006], *Software Life Cycle Management* (SLIM), *Function Point* (FP) [Sachan e Kushwaha 2018]. O COCOMO básico é um modelo amplamente conhecido e aceito, porém não provê estimativa realística em problemas relacionados a estimativa de esforço em *software*s complexos da atualidade [Sachan et al. 2016]. Em pesquisas anteriores, técnicas que empregam meta-heurísticas como Algoritmo Genético (AG), Rede Neural (RN), lógica *Fuzzy* entre outras foram utilizadas para proporcionar boa estimativa na função de otimização dos parâmetros do COCOMO básico [Sheta 2006]. Entre as meta-heurísticas abordadas na literatura o AG proporciona resultados promissores em relação obtenção de parâmetros que geram estimativa de esforço próximo ao esforço real, que é o esforço de fato empregado e mensurado após o desenvolvimento, denotado em termos de homem-mês [Singh e Misra 2012, Sheta 2006, Sachan et al. 2016]. Esta pesquisa propõe um AG para otimizar os parâmetros A e B do COCOMO básico. Para averiguar a acurácia por meio da comparação de resultados, é utilizado a base de dados de dezoito projetos da NASA que fornece *Kilo Line of Code* (KLOC), que se refere a valores denotados em termos de mil linhas de código, e suas respectivas estimativas de esforço mensurado.

2. Revisão Sistemática

O escopo da pesquisa foi estabelecido para responder a seguinte pergunta: Um Algoritmo Genético pode apresenta melhores resultados na otimização dos parâmetros do COCOMO básico em relação a outras técnicas utilizando configurações diferentes dos Algoritmos Genéticos encontradas na revisão bibliográfica efetuada?

2.1 Planejamento

Inicialmente foi selecionada a base de pesquisa do material bibliográfico. A definição de uma boa fonte de pesquisa é essencial para a revisão sistemática da literatura eficiente. Por meio da fonte de busca foi avaliado, a partir dos resultados retornados, o norteamento da lógica utilizada para criar a *string* de busca. A qualidade da *string* de busca também é importante pois, ela é o resumo da pergunta inicial da pesquisa, expressa em palavras chave que visam retornar material bibliográfico capaz de auxiliar o autor a desenvolver o trabalho. Foi selecionado o Google Acadêmico para efetuar as buscas, pois o mesmo detém vasto acervo de material bibliográfico favorecendo a abrangência da pesquisa. A definição do PICOC é imprescindível para estruturar o pensamento científico durante a pesquisa. A Tabela 1 abaixo descreve o PICOC.

Tabela 1. PICOC.

População	Engenharia de *Software*, Gestão de projetos, Estimativa de Esforço
Intervenção	COCOMO básico
Comparação	Modelos Metaheurísticos
Resultado	Obter estimativa precisa de esforço no desenvolvimento de *software*
Contexto	Projetos de *Software*

Em sequência, a partir da definição do PICOC é possível usá-lo para extrair palavras chave que representam o cerne da pesquisa. As palavras chave são estruturadas em uma *string* de busca com o auxílio de operadores lógicos (AND, OR). Esta estrutura tem o objetivo de retornar como resultado, um conjunto de artigos que em seu conteúdo possuam as palavras chave ligadas ao tema central. Na Tabela 2 é apresentada a base e *string* de busca.

Tabela 2. Base e *String* de busca.

Base	*String* de busca
Google Acadêmico	COCOMO AND (”function point” OR ”ponto de função”)AND(”Meta-heurística”OR ”Metaheuristic”OR ”Metaheurística”)OR (”heurística”OR ”heuristic”) AND (”Estimativa de Esforço”OR ”Effort Estimation”) AND (Optimization OR Otimização)

Após a definição da *string* de busca foi necessário estipular os critérios de inclusão e exclusão do material retornado a partir da *string*. Os critérios ajudam o pesquisador a selecionar de maneira mais crítica o material analisado com o objetivo de permanecer dentro do escopo do tema trabalhado.

Os critérios de inclusão definidos para esta pesquisa foram:

- Publicações do tipo artigo ou livro;
- Publicações da área de ciência da computação;
- Abordagem do mesmo tipo de problema.

Os critérios de exclusão definidos para esta pesquisa foram:

- Publicações que não são do tipo artigo ou livro;
- Publicações que não são da área de ciência da computação;
- Não abordagem do mesmo tipo de problema.

A revisão sistemática da literatura obteve um total de 188 artigos encontrados e analisados. Dos artigos analisados, 21 se mostraram aptos a fazer parte do trabalho proposto. O resumo da revisão sistemática da literatura é apresentado na Tabela 3.

Tabela 3. Resumo da revisão bibliográfica.

Artigos encontrados	188
Artigos eliminados após a leitura do título	168
Artigos eliminados após a leitura do *abstract*	4
Artigos selecionados pela metodologia *Snowballing*	4
Artigo cedido pelo autor Rohit Kumar Sachan	1
Total de artigos selecionados	21

3. Trabalhos Relacionados

A busca pela otimização dos parâmetros do COCOMO Básico tem gerado muitos trabalhos nos últimos anos [Chirra et al. 2019]. É importante ressaltar que o emprego de grande esforço em pesquisa reflete a importância que tal tema representa para o projeto e desenvolvimento de *software*. Muitos algoritmos genéticos com técnicas diferentes foram utilizados para superar o problema de acurácia apresentado pelo COCOMO básico.

Em 2006, Sheta propôs um AG para otimizar os parâmetros do COCOMO básico que obteve bons resultados para a época [Sheta 2006].

Outros bons resultados apresentados por AG em pesquisas relacionadas ao tema foram documentados nos últimos anos [Sheta 2006, Singh e Misra 2012, Sachan et al. 2016, Dhiman e Diwaker 2013].

Durante o levantamento bibliográfico foi possível observar a relevância dos resultados obtidos por AG em relação a outras técnicas como o Particle *Swarm Optimization* (PSO) [Singh e Misra 2012, Bozhikova e Stoeva 2014], que é uma técnica estocástica inspirada no comportamento social de aves [Sehra et al. 2017, Sachan e Kushwaha 2018]. [Sehra et al. 2017] aborda metodologias e técnicas de Computação Evolucionária (CE) para estimativa de esforço de *software*. Em sua pesquisa Sehra compara o ACO, o BCO e o PSO. As equações de avaliação utilizadas foram *Mean Magnitude of Relative Error* (MMRE) e *Root Mean Square Error* (RMSE). Os resultados mostraram bons resultados utilizando MMRE em relação ao esforço real.

Pesquisas na área de estimativa de esforço exibem uma extensa gama de abordagens aplicadas aos problemas. Entre tais abordagem de pesquisa é possível citar as técnicas *Ant Colony Optimization* (ACO) [Sehra et al. 2017, Bozhikova e Stoeva 2014], *Bee Colony Optimization* (BCO) [Dizaji et al. 2014, Sehra et al. 2017] e *Bat Algorithm* (BA) [Asghari Agcheh Dizaj e Soleimanian Gharehchopogh 2018].

No trabalho apresentado por Sachan e Kushwaha em 2018 os autores comparam os resultados obtidos por dois algoritmos, o AG [Sheta 2006] e o PSO [Sachan e Kushwaha 2018] com o resultado alcançados pelo COCOMO básico onde apesar do AG deter bom resultado o PSO expressou maior êxito na busca por estimativa de esforço em relação às demais técnicas apresentadas no trabalho [Sachan e Kushwaha 2018].

4. COCOMO Básico

O COCOMO Básico, é um modelo amplamente aceito para estimativa de esforço no desenvolvimento de software, representado na equação 1. O modelo foi proposto por Barry W. Boehm em 1981 e foi construído baseando-se em 63 projetos de *software* [Sachan et al. 2016]. O modelo utiliza o KLOC para fazer uma relação entre o número de linha e o esforço empregado para desenvolver tais linhas.

$$E = A * (KLOC)^{B} \quad (1)$$

O esforço estimado pelo modelo é dado em homem/mês. O COCOMO Básico possui três variantes, são elas: Orgânica; Semi-Destacada e Embarcada [Nandal e Sangwan 2016]. Na Tabela 4, é possível visualizar cada uma das variantes do COCOMO básico assim como suas variáveis A e B com os respectivos valores propostos por Barry W. Boehm [Sachan et al. 2016].

Tabela 4. Modelo do COCOMO básico.

Nome do Modelo	Tamanho do Projeto	Esforço
Modelo Orgânico	Menos que 50 KLOC	$E = 2.4 * (KLOC)^{1.05}$
Modelo Semi-Destacado	50-300 KLOC	$E = 3.0 * (KLOC)^{1.20}$
Modelo Embarcado	Acima de 300 KLOC	$E = 3.6 * (KLOC)^{1.12}$

O principal problema apresentado pelo COCOMO Básico é o fato de não apresentar estimativas realísticas para *software*s complexos encontrados no cenário atual. Portanto, nas últimas décadas vários trabalhos têm proposto modelos e técnicas com o objetivo de superar esta insuficiência exposta pelo COCOMO Básico [Sachan et al. 2016].

5. Algoritmo Genético

Algoritmo Genético é um algoritmo de busca evolucionária que pertence ao grupo de algoritmos inspirados na natureza. De acordo com [Chalotra et al. 2015] foi proposto por John Holland que abstraiu os conceitos da evolução das espécies [Chalotra et al. 2015, Sachan et al. 2016].

Esta Meta-Heurística utiliza procedimentos iterativos para criar "populações" formados por "indivíduos". Os "indivíduos" são avaliados por uma função chamada "*fitness*" e a partir de então inicia-se a seleção dos melhores avaliados para que estes possam ir para o próximo passo, o "*crossover*", onde ocorre o cruzamento dos mais aptos gerando "filhotes" para a próxima "geração". Os "filhotes" têm uma probabilidade de ter "mutação" [Chalotra et al. 2015, Sheta 2006].

6. Base de dados.

Para calcular os esforços estimados por meio do KLOC foi utilizado a base de dados de projetos de *software*s da NASA [Bailey e Basili 1981]. Esta base, apresentada na Tabela 5, é amplamente apresentada em trabalhos anteriores [Sheta 2006, Chalotra et al. 2015, Sachan et al. 2016, Maleki et al. 2014] como a fonte dos dados utilizados para estimativa. A base foi proposta a partir de um estudo publicado em 1981 por J. W. Bailey [Bailey e Basili 1981] e contém o número de cada projeto, no total de 18 projetos, o KLOC, citado anteriormente.

Tabela 5. Base de dados de Projetos de *Software* da NASA.

Nº do Projeto	KLOC	Esforço Real
1	90.2000	115.8000
2	46.2000	96.0000
3	46.5000	79.0000
4	54.5000	90.8000
5	31.1000	39.6000
6	67.5000	98.4000
7	12.8000	18.9000
8	10.5000	10.3000
9	21.5000	28.5000
10	3.1000	7.0000
11	4.2000	9.0000
12	7.8000	7.3000
13	2.1000	5.0000
14	5.0000	8.4000
15	78.6000	98.7000
16	9.7000	15.6000
17	12.5000	23.9000
18	100.8000	138.3000

7. Função Fitness

A função usada para qualificar cada cromossomo, chamada de *fitness* [Harman e Jones 2001], possibilita diferenciar e escolher os mais aptos para que possam participar do processo de cruzamento durante a execução do AG.

Por envolver muitas tentativas os algoritmos utilizados em problemas de busca podem ser lentos. Neste trabalho, foi utilizado os parâmetros do COCOMO básico com o intuito de otimizá-los. O COCOMO básico possui dois parâmetros: A e B. Apesar do universo de soluções se tornar grande pelo fato dos valores das variáveis A e B, quando relacionados na equação, proporcionar demasiadas soluções possíveis, este fato não implicou em menor eficiência. Algoritmo é capaz de obter boas soluções em alguns segundos [Harman e Jones 2001].

A função utilizada é o *Mean Absolute Error* (MAE) mostrada na Equação 2. MAE é a média do somatório das diferenças entre o esforço e esforço estimado de cada projeto. A função MAE será utilizada como o parâmetro de avaliação de qualidade do indivíduo,

chamado de *fitness*, para otimizar os parâmetros do COCOMO Básico a fim de estimar o esforço de *software* com maior eficiência. A função será calculada a partir do conjunto de esforços mensurados da base de dados de 18 projetos da NASA [Bailey e Basili 1981] com os respectivos KLOC's e esforços reais [Sachan et al. 2016].

$$\frac{1}{n}\sum_{i=1}^{n} |EsforçoReal_i - EsforçoEstimado_i| \quad (2)$$

8. Trabalho Proposto

O AG proposto tem o objetivo de otimizar os parâmetros A e B do COCOMO Básico. Existem muitas maneiras de implementar as configurações do AG [Sheta 2006, Chalotra et al. 2015, Sachan e Kushwaha 2018, Maleki et al. 2014]. A configuração utilizada no AG proposto é demonstrada na Tabela 6.

Tabela 6. Configuração de AG.

Mecanismo de Seleção	Seleção Aleatória com Elitismo
Tipo de Cruzamento	Recombinação em quatro pontos
Tipo de Mutação	Multi-Pontos
Tamanho da população	10
Máximo de gerações	100
Domínio de busca para A	0:10
Domínio de busca para B	0:10

8.1 Representação

Como representação do problema foi utilizado um conjunto de números inteiros em cadeia. Esta representação do problema é definida nas denominações dos componentes do algoritmo genético como cromossomo. O cromossomo é a representação da solução que está sendo buscada. Indivíduos são compostos de um cromossomo e um *fitness*. Os indivíduos compõem uma população. Na abordagem do trabalho proposto foi utilizada populações com dez indivíduos durante todas as gerações. Uma geração representa o momento atual da população onde durante as iterações do algoritmo genético as gerações seguintes recebem uma população com indivíduos filhos dos indivíduos melhor avaliados na geração passada.

8.2 Cruzamento

No cruzamento, os cromossomos dos indivíduos são combinados a fim de criar filhotes. O indivíduo filho possui cromossomo com características herdadas dos pais. A Figura 3 apresenta os filhos resultantes do cruzamento demonstrado na Figura 2. Tais características são chamadas de genes e tornam cromossomos diferentes entre si.

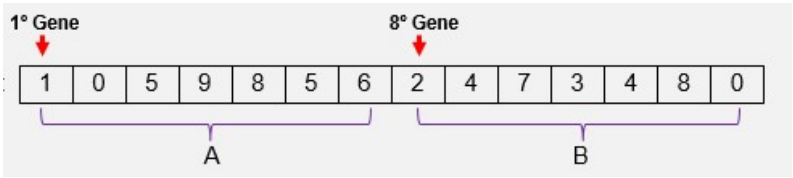

Figura 1. Cromossomo.

No trabalho proposto os genes são representados por números inteiros que variam de 0 a 9 presentes em 14 posições consecutivas em um vetor. O cromossomo é dividido em duas partes, “A” e “B”, referentes aos dois parâmetros do COCOMO básico. A parte “A” tem início no 1º gene e termina no 7º gene. A parte ”B” tem início no 8º gene e termina no 14º gene. Desta maneira “A” e “B” são representados no mesmo cromossomo. A Figura 1 apresenta um cromossomo e as posições dos genes contidos nele. Cada parte se tornará um número fracionário onde o 1º gene de cada parte foi denominado como a parcela inteira e a partir do 2º gene as parcelas fracionárias. Os pontos de troca de informação genética durante o cruzamento foram previamente estabelecidos. Tais pontos possibilitam a troca de dois genes consecutivos entre os pares cromossomos dos indivíduos pais.

A utilização de quatro pontos para a execução do cruzamento é baseada no fato do cromossomo possuir duas partes distintas que equivalem ao espaço de busca do parâmetro A e do parâmetro B. Para cada parte do cromossomo é atribuído dois pontos para o cruzamento. Os pontos foram dimensionados para efetuar troca genéticas de duas posições entre os cromossomos pais. Cada uma das partes, A e B, apresentam dois pontos para efetuar a troca genética. Os pontos foram criados para fazer trocas genéticas entre genes que representam os dígitos mais significativos, próximos da virgula, e menos significativos distantes da vírgula. Na Figura 2, os pontos referentes a A e B são apresentados. Tais pontos tem o objetivo de, aleatoriamente, se mover uma posição para a direita ou para a esquerda a partir de suas respectivas posições centrais e assim permitir a troca de genes de posições diferentes em a cada cruzamento. A Figura 2 mostra os pontos de cruzamento posicionados em suas posições centrais de movimentação. Este fato tornar possível a diversificação dos genes explorando melhor o espaço de busca.

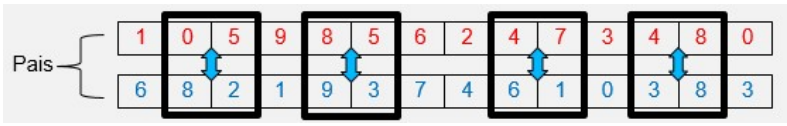

Figura 2. Indivíduos pais.

Figura 3. Indivíduos filhos.

Foram estipulados 4 pontos ao longo do cromossomo. Estes pontos se movimentam aleatoriamente três posições à direita ou à esquerda. Cada parte possui dois pontos onde acontecerão as trocas. Desta maneira é garantido que nem toda troca de informação genética será feita nos mesmos pontos dos cromossomos pais. Esta técnica tem como

objetivo aumentar a diversidade de genes nos cromossomos do indivíduo filho. A Figura 2 apresenta dois indivíduos e os pontos de troca genética.

8.3 Mutação

A mutação do cromossomo consiste em adicionar um gene completamente novo em alguma posição previamente estabelecida ou aleatória no cromossomo. O gene submetido a mutação não tem relação com nenhum cruzamento entre indivíduos.

Existem maneiras diferentes de aplicar a mutação. No trabalho proposto foi aplicada a mutação em 4 faixas fixas onde cada faixa compreende 3 posições consecutivas no cromossomo. A 1º faixa compreende o 1º gene até o 3º gene, a 2º faixa compreende o 4º gene até o 6º gene. A 1º e a 2º faixa pertencem à parte "A" do cromossomo. A 3º faixa compreende o 8º gene até o 10º gene e a 4º faixa compreendo o 11º até o 13º gene.

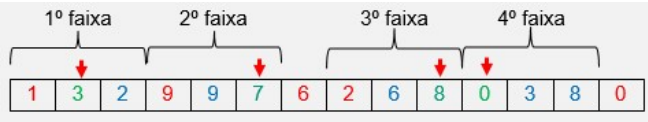

Figura 4. Faixa onde ocorrem as mutações.

A 3º e 4º faixa pertencem a parte "B" do cromossomo. Um gene é escolhido aleatoriamente para sofrer mutação em cada faixa. Existe 20\% de chance de um cromossomo sofrer mutação. A Figura 4 representada as faixas onde acontecem as mutações em posições aleatórias.

8.4 Algoritmo Proposto

Para otimizar os parâmetros COCOMO Básico foi utilizado os seguintes passos:

Passo 1: Gerar aleatoriamente a primeira geração com Indivíduos que possuem cromossomos como na configuração da Figura 5.

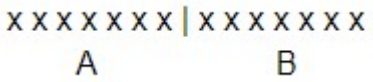

Figura 5. Partes do cromossomo relativas aos parâmetros A e B do COCOMO básico.

As variáveis A e B são expressas por 7 genes cada. Onde o primeiro gene de cada variável é a parte inteira do valor representado e os genes restantes representam a parte fracionada.

Passo 2: Calcular o esforço estimado utilizando os valores de KLOC dos respectivos projetos encontrados na base de dados da NASA.

Passo 3: Calcular o fitness de cada Indivíduo contido na população. Nesta fase é feito o cálculo do MAE utilizando o esforço estimado no passo 2. O fitness do indivíduo será a média da soma do módulo das diferenças entre esforço estimado e esforço real. O valor fitness atribuído ao indivíduo faz parte do método de seleção utilizado neste modelo de Algoritmo Genético proposto. A seleção é feita escolhendo-se os melhores indivíduos.

O valor do *fitness* é o erro médio do obtido pelo cálculo da estimativa de esforço utilizando os parâmetros A e B contidos no cromossomo do indivíduo. Neste caso considerando que o valor ideal de erro é 0, quanto menor o valor do fitness, melhor é o resultado proporcionado pelo cromossomo do indivíduo. A partir do cálculo dos erros é possível "guiar" as soluções obtidas em direção a resultados mais precisos.

Passo 4: A condição de cem gerações sem melhora foi escolhida como condição de parada. Existem muitas variações em relação a condição de parada. A condição de parada está relacionada as configurações escolhidas para o Algoritmo Genético. As configurações do AG são responsáveis pelo desempenho do mesmo diante determinado problema. Se o método de cruzamento não for escolhido adequadamente considerando as peculiaridades do problema abordado pode apresentar uma melhora lenta. A melhora lenta pode implicar em não atingir bons resultados dentro do critério de parado escolhido. Este critério foi escolhido baseado na literatura onde foi observado que 100 gerações sem melhora foram suficientes para gerar bons resultados.

Passo 5: Selecionar 50% da população cujos indivíduos tiveram as melhores classificações. No trabalho proposto, 50% dos indivíduos equivalem a cinco indivíduos.

Esta escolha empírica foi utilizada partindo do suposto que, menos que 5 indivíduos em uma população de tamanho 10, podem conter pouco material genético variado para criar gerações distintas das anteriores. Pouco material genético implica em menos locais de busca no universo de soluções possíveis. Neste caso, 50% de uma população de 10 indivíduos mostrou-se suficiente para manter a diversidade genética durante os cruzamentos ao longo das gerações.

Passo 6: Cruzar aleatoriamente os indivíduos selecionados no Passo 5. Foi usada a recombinação em quatro pontos não uniformes como a estratégia de cruzamento. Os filhotes, resultantes dos cruzamentos e o indivíduo melhor classificado, conforme o elitismo, foram alocados na população da geração seguinte. O indivíduo considerado melhor de sua geração segue para a próxima geração sem chances de sofrer mutação, contudo, antes de seguir para próxima geração, o mesmo participa do processo de cruzamento aleatório com os indivíduos considerados mais “aptos” como descrito no passo 5 onde, os filhos resultantes desta etapa têm a possibilidade de sofre mutações.

O cruzamento aleatório pode ocasionar durante as gerações a perda de cromossomos que contem genes potencialmente bons em relação a posição onde se encontra no cromossomo. Pelo fato do cruzamento aleatório possibilitar que indivíduos não tão aptos cruzem com indivíduos bem classificados, há o risco de que haja uma demora na melhora das soluções encontradas pelo Algoritmo Genético. A solução encontrada para tal fato foi combinar o Elitismo como o cruzamento aleatório. O elitismo envia os indivíduos mais apto para a próxima geração fazendo com que a perda de genes bons não aconteça. Mesmo que os cruzamentos da geração atual não sejam tão bons a ponto de gerar filhos mais aptos o elitismo garante que os resultados bons não se percam e na próxima geração ainda existe a possibilidade de melhorar.

Passo 7: Um indivíduo na população tem 20% de chances de ser selecionado para sofrer mutação. A mutação é configurada em 4 pontos no cromossomo. A mutação se torna importante pelo fato de que apenas o cruzamento entre indivíduos não possibilita busca por um amplo universo de soluções.

Durante o cruzamento, uma parte da informação genética de um indivíduo é transferido para outro com o objetivo de gerar novos filhos. O cromossomo dos filhos é fruto dessa transferência, portanto contém os mesmos genes dos pais. Quando a mutação ocorre, um gene é inserido em uma posição, que neste trabalho é aleatória, onde este gene não tem nenhuma relação com o cruzamento entre os pais do novo indivíduo. Este novo gene proporciona uma nova perspectiva de soluções pois, se este novo gene impactar positivamente nas soluções obtidas até o momento, o cromossomo que o carrega será selecionado para o cruzamento e passará o gene promissor para frente ou se for considerado o indivíduo mais apto, pela ação do elitismo, seguirá imediatamente para a próxima geração. A mutação é um dos fatores responsáveis pela obtenção de bons resultados no Algoritmo Genético.

Passo 8: A nova população está formada neste passo. Em seguida é iniciado o processo a partir do passo 2 repetidas vezes até que a condição de parada seja atingida e assim encerrar a execução algoritmo.

9. Resultados

A implementação do Algoritmo Genético utilizando o conjunto de estratégias como elitismo, seleção aleatória e mutação em quatro pontos foi o modelo proposto neste trabalho. O modelo proposto apresenta estimativa de esforço mais próximas ao esforço real que o COCOMO básico como pode ser observado na Figura 6. A Figura 8 apresenta os resultados obtidos pelo AG proposto neste trabalho.

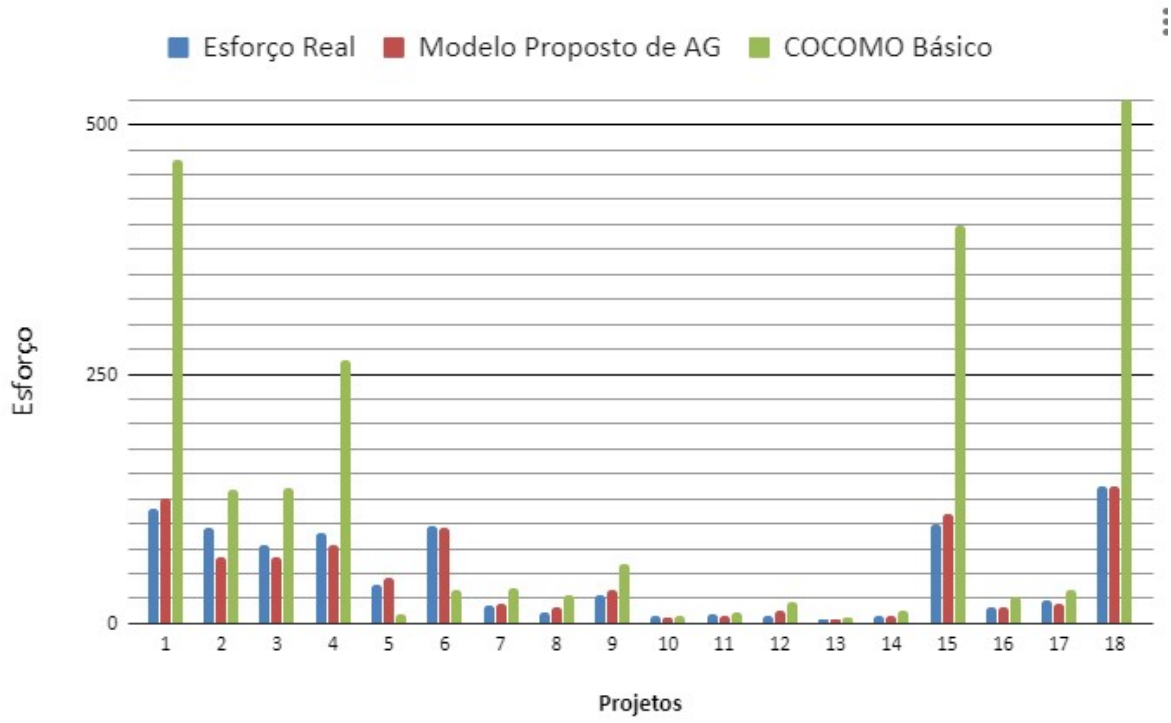

Figura 6. Comparação entre o COCOMO básico e o Modelo Proposto de AG.

Os artigos obtidos por meio da revisão bibliográfica trouxeram informação sobre resultados relevantes para esta pesquisa. Alguns trabalhos com metodologia parecida foram selecionados e serviram de base para este trabalho.

Em outras abordagens os autores apresentaram modelos para otimizar os parâmetros do COCOMO básico [Harman e Jones 2001]. A complexidade existente em projetos de *software* faz com que o modelo do COCOMO básico proposto por [Barry et al. 1981] apresente deficiência em estimar o esforço no desenvolvimento de *software* atualmente [Sachan et al. 2016]. A utilização de AG e outras técnicas algorítmicas são tema de diversos trabalhos nas últimas décadas.

Em 2006 foi apresentado por [Sheta 2006] um trabalho que testa o AG aplicado a três modelos do COCOMO para otimizar os parâmetros A e B no primeiro modelo, A, B e C no segundo modelo e A, B, C e D no terceiro modelo. Em 2011 [HARI e PVGD 2011] também utiliza *Particle Swarm Optimisation* (PSO) aplicado a três modelos do COCOMO para otimizar os parâmetros A e B no primeiro modelo; A, B e C no segundo modelo e A, B, C e D no terceiro modelo. [Sachan e Kushwaha 2018] utilizou um NIA, algoritmo inspirado na natureza que abstrai o comportamento anti-predador dos sapos para a otimizar os parâmetros A e B do COCOMO básico. Na Tabela 8 são apresentados os resultados dos modelos apresentados pelos autores citados acima e inclusive os resultados do AG proposto neste trabalho. Na Tabela 9 é possível observar detalhes sobre a solução produzida pelo AG. Na Tabela 10 é apresentado os resultados A e B dos respectivos modelos comparados neste trabalho.

Os erros obtidos por cada modelo em relação a estimativa de esforço de cada projeto de *software* da NASA são apresentados na Tabela 11.

Tabela 7. Resultado de Estimativa de Esforço do modelo proposto

Nº do Projeto	KLOC	Esforço Real	Modelo Proposto
1	90.2000	115.8000	124.713
2	46.2000	96.0000	66.9144
3	46.5000	79.0000	67.3186
4	54.5000	90.8000	78.0355
5	31.1000	39.6000	46.2989
6	67.5000	98.4000	95.2247
7	12.8000	18.9000	20.2669
8	10.5000	10.3000	16.8553
9	21.5000	28.5000	32.8381
10	3.1000	7.0000	5.41618
11	4.2000	9.0000	7.18496
12	7.8000	7.3000	12.7822
13	2.1000	5.0000	3.76958
14	5.0000	8.4000	8.45061
15	78.6000	98.7000	109.718
16	9.7000	15.6000	15.657

17	12.5000	23.9000	19.8245
18	100.8000	138.3000	138.298

Tabela 8. Comparação entre esforços estimados.

Nº do Projeto	Esforço Real	Modelo Proposto 2019	NIA 2018	PSO 2011	AG 2006	COCOMO básico
1	115.8000	124.713	125.4714	125.7302	131.9154	464.4646
2	96.0000	66.9144	69.5447	70.8350	80.8827	134.3029
3	79.0000	67.3186	69.9428	71.2293	81.2663	135.2187
4	90.8000	78.0355	80.4548	81.6179	91.2677	264.1711
5	39.6000	46.2989	49.0527	50.4476	60.5603	88.6359
6	98.4000	95.2247	97.1622	98.0537	106.7196	335.6924
7	18.9000	20.2669	22.4183	23.5607	31.6447	34.8965
8	10.3000	16.8553	18.8249	19.8799	27.3785	28.3439
9	28.5000	32.8381	35.4207	36.7575	46.2352	60.155
10	7.0000	5.41618	6.4183	6.9828	11.2212	7.873
11	9.0000	7.18496	8.3897	9.0602	14.0108	10.8299
12	7.3000	12.7822	14.4834	15.4064	22.0305	20.7449
13	5.0000	3.76958	4.5523	5.0000	8.4406	5.2305
14	8.4000	8.45061	9.7843	10.5215	15.9157	13.0056
15	98.7000	109.718	111.1258	111.7296	119.2850	398.1021
16	15.6000	15.657	17.554	18.5737	25.8372	26.0808
17	23.9000	19.8245	21.9543	23.0863	31.1008	34.0382
18	138.3000	138.298	138.3902	138.3002	143.0788	526.0137

Tabela 9. Resultados do AG proposto

Operadores	Saídas
A	1.889920
B	0.930576
Cromossomo mais apto	188992009305761
MAE	6.10521
Total de Gerações	446

Tabela 10. Representações de soluções de A e B apresentadas pelos modelos

Modelo	A	B
AG	4.9067	0.7311
PSO with inertia weight	2.6463	0.8576
NIA	2.3661	0.88201
Proposto	1.889920	0.930576

Tabela 11. Comparação entre os erros das estimativas de esforço de cada modelo

Nº do Projeto	Modelo Proposto 2019	NIA 2018	PSO 2011	AG 2006	COCOMO básico 1981
1	8.913	9.6714	9.9302	16.1154	348.6646
2	29.0856	26.4553	25.165	15.1173	38.3029
3	11.6814	9.0572	7.7707	2.2663	56.2187
4	12.7645	10.3452	9.1821	0.4677	173.3711
5	6.6989	9.4527	10.8476	20.9603	49.0359
6	3.1753	1.2378	0.3463	8.3196	237.2924
7	1.3669	3.5183	4.6607	12.7447	15.9965
8	6.5553	8.5249	9.5799	17.0785	18.0439
9	4.3381	6.9207	8.2575	17.7352	31.655
10	1.58382	0.5817	0.0172	4.2212	0.873
11	1.81504	0.6103	0.0602	5.0108	1.8299
12	5.4822	7.1834	8.1064	14.7305	13.4449
13	1.23042	0.4477	0	3.4406	0.2305
14	0.05061	1.3843	2.1215	7.5157	4.6056
15	11.018	12.4258	13.0296	20.585	299.4021
16	0.057	1.954	2.9737	10.2372	10.4808
17	4.0755	1.9457	0.8137	7.2008	10.1382
18	0.002	0.0902	0.0002	4.7788	387.7137

A função MAE usada como função fitness pelo AG, é utilizada para classificar o cromossomo. Cromossomo é uma representação da solução encontrada pelo AG, ou seja, os valores de A e B. MAE prove a média absoluta dos erros apresentados entre o esforço estimado e o esforço real. Um valor fornecido por MAE mais próximo de zero representa uma solução mais precisa em relação ao objetivo real. A Figura 7 apresenta a relação de gerações e os respectivos melhores fitness associados, expressos em um gráfico da curva de convergência fornecida pelo Algoritmo Genético proposto.

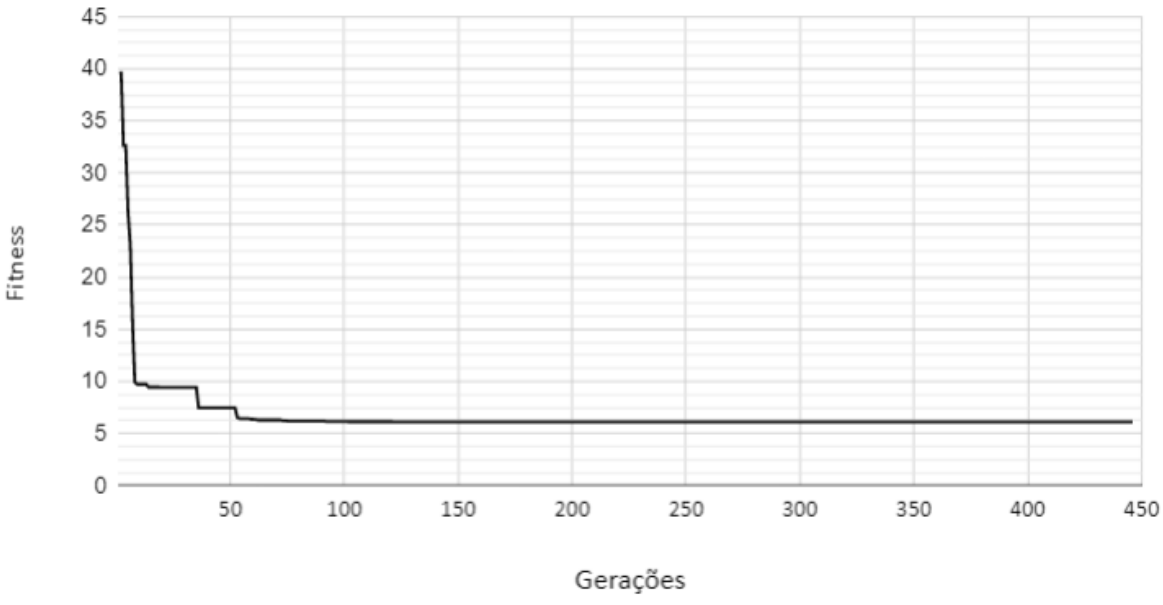

Figura 7. Curva de convergência do apresentado pelo Algoritmo Genético proposto.

A Figura 8 mostra os resultados em termos de MAE dos modelos abordados. O modelo proposto obteve o valor de 6.10519 de erro médio absoluto, demonstrando maior acurácia em relação aos modelos comparados.

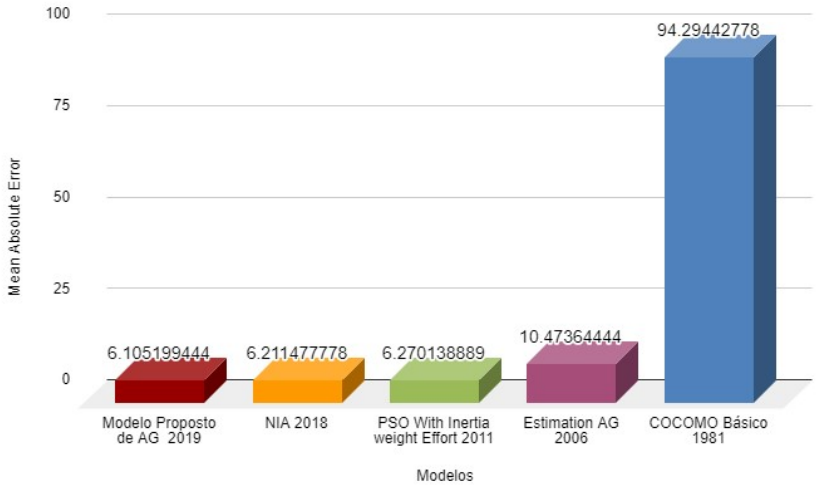

Figura 8. Resultado em termos de MAE apresentados pelos modelos apresentados

10. Conclusão

Durante a construção deste trabalho, foi possível observar a ampla variedade de empregabilidade de técnicas de busca e otimização em problemas clássicos de Engenharia de *Software*. *Search-Based Software Engineering* é a área de estudo que aplica técnicas de otimização a problemas de Engenharia de *Software*. Esta é uma área que expande significantemente juntamente com necessidade de gerenciar projetos de desenvolvimento de *software* cada vez mais complexos.

Neste estudo foi proposto um AG para otimizar os parâmetros A e B do COCOMO básico. Muitos trabalhos na área foram desenvolvidos nas últimas décadas utilizando técnicas algorítmicas variadas [Chirra et al. 2019]. Precisão na estimativa de esforço no desenvolvimento de *software* e fornece segurança e melhor gerenciamento dos recursos relacionados ao projeto de *software* [Harman e Jones 2001]. O AG proposto fornece resultados mais precisos em relação aos modelos apresentados. A acurácia demonstrada pode ser relacionada à configuração apresentada pelo AG.

O resultado mostra que o AG proposto apresenta em termos de MAE melhoria de 93.5253% comparado ao COCOMO Básico [Barry et al. 1981], 41.7089% de melhoria comparado a outro modelo de AG proposto por [Sheta 2006], 2.6305% de melhoria comparado ao modelo proposto por [HARI e PVGD 2011] de PSO e 1.7109% de melhoria comparado ao modelo proposto por [Sachan e Kushwaha 2018] de *Nature-Inspired Algorithm* (NIA).

Como trabalhos futuros a aplicação de outras técnicas como Rede Neural e *Simulated Annealing* podem ser abordadas com o objetivo de ampliar o conhecimento relacionado ao tema. Outra possibilidade de abordagem pode ser a utilização de meta-heurísticas como o Algoritmo Genético para otimizar os parâmetros do COCOMO II [Dhiman e Diwaker 2013]. A construção de uma ferramenta que auxilia na estimativa de esforço no desenvolvimento de software utilizando o modelo proposto é um exemplo de trabalho futuro onde pode ser aplicado na prática os conceitos apresentados e discutidos nesta pesquisa.

Referências

Asghari Agcheh Dizaj, S. and Soleimanian Gharehchopogh, F. (2018). A new approach in soft- ware cost estimation by improving genetic algorithm with bat algorithm. *Journal of Computer & Robotics*, pages 31–44.

Bailey, J. W. and Basili, V. R. (1981). A meta-model for software development resource expen- ditures. In *Proceedings of the 5th international conference on Software engineering*, pages 107–116. IEEE Press

Barry, B. et al. (1981). Software engineering economics. *New York*, 197.

Bozhikova, V. T. and Stoeva, M. T. (2014). Search-based approach for software cost estimation. *ANNUAL JOURNAL OF ELECTRONICS.*

Chalotra, S., Sehra, S., Brar, Y., and Kaur, N. (2015). Tuning of cocomo model parameters by using bee colony optimization. *Indian Journal of Science and Technology*, 8:1.

Chirra, S. M. R., Reza, H., et al. (2019). A survey on software cost estimation techniques. *Journal of Software Engineering and Applications*, 12:226.

Dhiman, A. and Diwaker, C. (2013). Optimization of cocomo ii effort estimation using genetic algorithm. *American International Journal of Research in Science, Technology, Engineering & Mathematics*, 3.

Dizaji, Z. A., Ahmadi, R., Gholizadeh, H., and Gharehchopogh, F. S. (2014). A bee colony opti- mization algorithm approach for software cost estimation. *International Journal of Computer Applications*, 104.

Ferrucci, F., Harman, M., and Sarro, F. (2014). Search-based software project management. In *Software Project Management in a Changing World*, pages 373–399. Springer.

HARI, C. and PVGD, P. R. (2011). Software effort estimation using particle swarm optimization with inertia weight. *International Journal of Software Engineering (IJSE)*, 2(4):87–96.

Harman, M. and Jones, B. F. (2001). Search-based software engineering. *Information and software Technology*, 43(14):833–839.

Kaur, M. (2018). Estimation of effort using nature inspired optimization techniques. *International Journal of Academic Research and Development*, 3(1).

Kaushik, A., Verma, S., Singh, H. J., and Chhabra, G. (2017). Software cost optimization inte- grating fuzzy system and coa-cuckoo optimization algorithm. *International Journal of System Assurance Engineering and Management*, 8:1461–1471.

Maleki, I., Gharehchopogh, F. S., Ayat, Z., and Ebrahimi, L. (2014). A novel hybrid model of scatter search and genetic algorithms for software cost estimation. *Magnt Research Report*, 2(6):359–371.

Michalewicz, Z. (2013). *Genetic algorithms+ data structures= evolution programs*. Springer Science & Business Media.

Nandal, D. and Sangwan, O. P. (2016). A survey report on various software estimation techniques and practices. *International Science Press*.

Sachan, R. K. and Kushwaha, D. S. (2018). New nia based approach for optimizing basic co- como model. *5th International Conference on Computing for Sustainable Global Develop- ment,INDIACOM*, pages 1242–1247.

Sachan, R. K., Nigam, A., Singh, A., Singh, S., Choudhary, M., Tiwari, A., and Kushwaha, D. S. (2016). Optimizing basic cocomo model using simplified genetic algorithm. *Procedia Compu- ter Science*, 89:492–498

Sehra, S. K., Brar, Y. S., and Kaur, N. (2017). Evolutionary computing techniques for software effort estimation. *International Journal of Computer Science & Information Technology*.

Sheta, A. F. (2006). Estimation of the cocomo model parameters using genetic algorithms for nasa software projects. *Journal of Computer Science*, 2:118–123.

Singh, B. K. and Misra, A. (2012). Software effort estimation by genetic algorithm tuned parame- ters of modified constructive cost model for nasa software projects. *International Journal of Computer Applications*, 59.