UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
FACULDADE DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Adriano Silva Barreto
Thales Silva de Sousa

Análise comparativa entre novos operadores genéticos incluídos no *Framework Evolutionary Algorithms*

Belém – Pará

2019

Adriano Silva Barreto
Thales Silva de Sousa

Análise comparativa entre novos operadores genéticos incluídos no *Framework Evolutionary Algorithms*

Trabalho de Conclusão de Curso apresentado como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação. Universidade Federal do Pará. Faculdade de Computação. Instituto de Ciências Exatas e Naturais

Orientador: Profº. Drº. Claudomiro de Souza de Sales Junior
Coorientador: Bel. George Tassiano de Melo

Belém – Pará
2019

Adriano Silva Barreto
Thales Silva de Sousa
Análise comparativa entre novos operadores genéticos incluídos no *Framework Evolutionary Algorithms*/ Adriano Silva Barreto
Thales Silva de Sousa. – Belém – Pará, 2019-
75 p. : il. (algumas color.) ; 30 cm.

Orientador: Profº. Drº. Claudomiro de Souza de Sales Junior

Coorientador: Bel. George Tassiano de Melo

Trabalho de Conclusão de Curso – Universidade Federal do Pará – UFPa
Instituto de Ciências Exatas e Naturais – Icen
Faculdade de Computação – Facomp, 2019.
1. Algoritmo Genético. 2. Otimização de Operador. 3. Estudo Analítico. I. Profº. Drº. Claudomiro de Souza de Sales Junior. II. Universidade Federal do Pará. III. Faculdade de Computação. IV. Análise comparativa entre novos operadores genéticos incluídos no *Framework Evolutionary Algorithms*.

CDU - - : - - - : - - - . -

Adriano Silva Barreto
Thales Silva de Sousa

Análise comparativa entre novos operadores genéticos incluídos no *Framework Evolutionary Algorithms*

Trabalho de Conclusão de Curso apresentado como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação. Universidade Federal do Pará. Faculdade de Computação. Instituto de Ciências Exatas e Naturais

Belém – Pará, _____ de _____ de 20_____.

Conceito: _____.

Profº. Drº. Claudomiro de Souza de Sales Junior
Orientador

Bel. George Tassiano de Melo
Coorientador

Profº. Drº. Dionne Cavalcante Monteiro
Avaliador

Profº. Mscº. Mauro Rodrigo Larrat Frota e Silva
Avaliador

Belém – Pará
2019

AGRADECIMENTOS

Agradeço primeiramente a Deus, aos meus pais, ao meu irmão e todos os outros membros da minha família, por serem os alicerces que me mantém firme.

Também quero agradecer aos amigos que compartilharam comigo essa jornada, Caio, Filipe, George, Ricardo, Giordana, Yvan, Lopes, Eliasqueci, Lopes, Escudeiro, Crispino, Paulo, Kaê, Roberto e dizer que eles contribuíram muito para o meu crescimento durante esse período.

Um especial agradecimento ao meu amigo Thales que desenvolveu este trabalho junto comigo.

Agradeço também a todo o corpo docente da Faculdade de Computação que me instruiu durante essa jornada, principalmente ao meu orientador Claudomiro Sales que é uma pessoa que ajudou muito em vários momentos da graduação.

Adriano Barreto

AGRADECIMENTOS

À minha família, em especial à minha avó Waldete e minha mãe Jacqueline, pelo apoio e suporte que sempre foi oferecido.

À todos os amigos que me ajudaram durante essa caminhada, em especial aos amigos Adriano, Ariel, Rafael, Thiago, Paulo, Felipe, Isabella, Brenda, Gabriel, George, Caio, Kaê, Guilherme, Yago e diversos outros amigos.

Ao meu orientador Prof Claudomiro pela oportunidade e paciência durante a elaboração deste trabalho.

Ao meu amigo que seguiu nessa jornada de TCC comigo, Adriano Barreto.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico, pela Bolsa de Iniciação Científica que me proporcionou o ambiente de pesquisa necessário para expandir meu conhecimento.

Thales Sousa

“ Um homem que foge de seu medo somente encontrou um atalho para encontrá-lo.”

J. R. R. Tolkien

RESUMO

Novos operadores para Algoritmos Genéticos estão sendo propostos diariamente pela comunidade acadêmica, para melhorar o desempenho dessa técnica. É necessário conhecer o desempenho dos mesmos afim de fazer bom uso dessas melhorias e conhecer suas limitação e pontos fortes. Este trabalho trata-se de uma análise comparativa sobre variantes de algoritmos genéticos que foram criados e implementados pela comunidade acadêmica. O objetivo desse estudo é realizar comparações entre as variantes de operadores genéticos para identificar as diferenças existentes de desempenho oferecido por eles. Os operadores genéticos que foram pesquisados neste trabalho são: o operador transgênico, o operador de diversidade de parasitas e o operador imune adaptativo baseado na entropia da informação. Tais operadores foram implementados e avaliados através de testes com funções multimodais. Uma análise foi feita entre os algoritmos genéticos com o objetivo de avaliar, se o algoritmo encontra a solução e a garantia de convergência. Algumas métricas que foram avaliadas nos operadores foram, a robustez para otimizar a função com uma dada tolerância de erro e uma análise de convergência. Foi considerado neste trabalho que a solução é encontrada de acordo com várias precisões definidas, sendo quando o erro é menor ou igual a 10^{-3}, 10^{-2} e 10^{-1}. Após os testes, a análise de desempenho feita entre os operadores implementados, mostrou que todos os operadores obtiveram bons resultados para as funções com uma convergência boa e o operador que obteve os melhores resultados foi o operador imune adaptativo.

Palavras-chave: Operadores Genéticos Variantes de Algoritmo Genético, Algoritmo Genético, Computação Evolucionária

ABSTRACT

New operators for Genetic Algorithms are being proposed daily by the academic community to improve the performance of this technique. It is necessary to know their performance in order to make good use of these improvements and to know their limitations and strengths. This work is a comparative analysis of variants of genetic algorithms that were created and implemented by the academic community. The purpose of this study is to perform comparisons between variants of genetic operators to identify the existing differences in performance offered by them. The genetic operators that were researched in this work are: the transgenic operator, the operator of parasite diversity and the adaptive immune operator based on information entropy. These operators were implemented and evaluated through tests with multimodal functions. An analysis was made among the genetic algorithms in order to evaluate if the algorithm finds the solution and the convergence guarantee. Some metrics that were evaluated in the operators were the robustness to optimize the function with a given error tolerance and a convergence analysis. It was considered in this work that the solution is found according to various defined precisions, where the error is less than or equal to 10^{-3}, 10^{-2} and 10^{-1}. After the tests, the performance analysis performed among the implemented operators showed that all the operators obtained good results for the functions with a good convergence and the operator that obtained the best results was the adaptive immune operator.

Keywords: Genetic Operators Variants of Genetic Algorithm, Genetic Algorithm, Evolutionary Computation

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AG	Algoritmo Genético
AE	Algoritmo Evolucionário
DP	Desvio Padrão
FEA	*Framework Evolutionary Algorithms*
IGAIE	*Immune Genetic Algorithm based on Information Entropy*
IDE	Ambiente de Desenvolvimento Integrado (*Integrated Development Environment*)
PSO	*Particle Swarm Otimization*
MuGA	*Multiset Genetic Algorithm*
SMuGA	*Symbiogenetic Multiset Genetic Algorithm*
KH	*Krill Herd*
TGA	*Transgenic Genetic Algorithm*
UFPA	Universidade Federal do Pará

SUMÁRIO

1 INTRODUÇÃO

1.1 Contexto

Computação evolucionária é uma área de pesquisa dentro da ciência da computação, e caracteriza-se como um tipo de computação que busca a sua inspiração do processo de evolução natural da biologia. O fundamento metafórico da computação evolucionária se relaciona com a evolução natural de um estilo em particular, através de tentativa e erro.(EIBEN; SMITH, 2003)

A inspiração através da biologia vem diretamente da teoria da evolução de Charles Darwin, na qual existe um ambiente que tem a capacidade de hospedar uma quantidade limitada de indivíduos, com os instintos básicos de sobrevivência e reprodução. A seleção natural aparece de forma inevitável, forçando a população a manter os indivíduos mais adequados ao ambiente. Sendo assim, a seleção natural acaba selecionando os indivíduos mais fortes e capazes de continuar sobrevivendo no ambiente de acordo com as condições que estão presentes no mesmo. A mutação ocorre de forma gradual e lenta, eliminando um atributo a um ponto que não afete mais a sobrevivência da espécie do indivíduo no ambiente.(EIBEN; SMITH, 2003)

No básico da computação evolucionária, será dado um ambiente com uma população de indivíduos que lutam por sobrevivência e atualização da sua espécie através de reprodução. Segundo (EIBEN; SMITH, 2003) "O *f*itness destes indivíduos é determinado pelo ambiente e é relacionado as chances de sobrevivência e multiplicação de tais indivíduos". Trazendo esse conceito para a resolução de problemas, cada nova geração de indivíduos será melhor que a anterior, então teremos uma coleção de soluções melhores e mais adaptada ao meio no decorrer do processo de evolução das soluções.

Tais conceitos da biologia na teoria da computação evolucionária seria explicado como uma população que pode ser identificada como vários pontos em um cenário, aonde cada ponto é um indivíduo realizando uma possibilidade de combinação de atributos. A evolução é o processo gradual que avança a população para as áreas de altas altitudes, suportada pela variação e seleção natural. Existem problemas que vários pontos encontrados são melhores que todas as suas soluções vizinhas, foi atribuído para cada um desses pontos o nome de ótimo local, e o mais alto destes valores é o ótimo global. (EIBEN; SMITH, 2003)

Transformando tais teorias para a prática de computação evolucionária, agora temos um Algoritmo Genético (AG). Este AG irá se basear na seguinte ideia comum: é dada uma população de indivíduos em um ambiente com recursos limitados, a competição

por tais recursos causa a seleção natural dos indivíduos, então acontecera um aumento na aptidão, ou $f$itness, da população de indivíduos. Tendo como base o $f$itness mais alto dos indivíduos, quanto mais alto, melhor, assim serão escolhidos os melhores candidatos para dar continuidade para a próxima geração. Isso será feito aplicando a recombinação e mutação para os mesmos, a recombinação é um operador que é aplicado para dois ou mais candidatos selecionados, produzindo um novo candidato. Este candidato terá o seu $f$itness avaliado e ira competir com os indivíduos mais velhos para um lugar na próxima geração.(EIBEN; SMITH, 2003)

De acordo com (AMARAL; HRUSCHKA, 2011) e (LIU; YANG; PAN, 2015), algoritmos genéticos, foram inventados por John Holland na década de 1960 e foram desenvolvidos por Holland e seus alunos e colegas na Universidade de Michigan nos anos 1960 e 1970. Em contraste com as estratégias de evolução e programação evolucionária, Holland teve como objetivo original estudar formalmente o fenômeno da adaptação à medida que ocorre na natureza, e desenvolver maneiras pelas quais os mecanismos de adaptação natural podem ser importados para sistemas de computador. Holland apresenta no livro Adaptação em Sistemas Naturais e Artificiais, o algoritmo genético como uma abstração da evolução biológica e deu uma teoria quadro de adaptação no âmbito do AG.

John Holland apresentou quatro operadores para os AGs, como: seleção, crossover, mutação e inversão. O operador de seleção escolhe esses cromossomos na população que será permitido reproduzir, e em geral os cromossomos com melhor aptidão serão escolhidos para produzir mais prole que os outros. Os operadores de $c$rossover são responsáveis pela troca de subpartes de dois cromossomos selecionados, imitando a recombinação biológica entre dois organismos de cromossomo único ("haploide"). O operador de mutação, por outro lado, muda aleatoriamente os valores do alelo em locais específicos no cromossomo. O operador de inversão, realiza uma inversão da ordem de uma seção contígua do cromossomo, reorganizando a ordem na qual os genes são dispostos.

As abordagens computacionais podem ser aplicadas na resolução de diferentes desafios da biologia, como identificação e análise de expressões gênicas, comparação de sequencias (DNA, RNA e proteínas), montagem de fragmentos, reconhecimento de genes, determinação da estrutura de proteínas e unificação da representação de gene e produto gênico atributos entre espécies. (AMARAL; HRUSCHKA, 2014)

Novos operadores sempre estão sendo sugeridos e criados na comunidade acadêmica, o motivo dessas novas criações, é principalmente por causa destes operadores possuírem técnicas novas ou melhorias de técnicas existentes de outros autores. Neste trabalho, serão apresentados três operadores variantes de algoritmo genético para resolução de problemas, tais operadores são: o operador transgênico, o operador imune adaptativo baseado na entropia da informação e o operador de diversidade de parasitas.

O primeiro operador é o Operador Genético Transgênico (TGA) é baseado na ideia de organismos modificados através da engenharia genética, aonde o material genético dos indivíduos é manipulado e adicionado de forma artificial no mesmo. O operador identifica genes relevantes dos indivíduos com valores específicos, e guarda estes valores para os genes da próxima geração de indivíduos e assim ocorre a adaptação de cada geração. Os testes realizados por (AMARAL; HRUSCHKA, 2014), mostraram que o operador é promissor em obter os mesmos resultados que outros operadores, mas com um número menor de gerações.

O segundo operador é o Operador Genético de Diversidade de Parasitas (SMuGA), este operador é utilizado no modelo do algoritmo genético (MuGa)(HEYWOOD; LICHODZIJEWSKI, 2010), que possui sim-biogênese, sendo isto o princípio de gerar uma nova geração de espécies através de integração genética de diferente organismos, que são chamados de simbiontes. O SMuGA possui como base os parasitas e os hospedeiros, no qual os parasitas representam soluções parciais enquanto os hospedeiros são as soluções completas, os parasitas tendem a se concentrar no máximo local, porém eles tendem a crescer e de acordo com o seu sucesso, propagar as soluções ótimas locais e desta forma prevenir a fuga e a possibilidade de se chegar no ótimo global.

O Operador Imune Adaptativo Baseado na Entropia da Informação (IGAIE) é o terceiro operador tratado neste trabalho, de acordo com o autor (LIU; YANG; PAN, 2015), a eficácia de operadores que são baseados em imunidade é algo problemático, e para isso o IGAIE foi proposto. Algoritmo imune é um algoritmo de otimização global com uma concentração de anticorpos baseado no sistema imune, para o controle da diversidade da população de anticorpos. Grandes concentrações de anticorpos convergem para a inibição, a entropia da informação tem o seu papel como indicador para mesurar a similaridade, ou afinidade, entre anticorpos e representar ao número de concentração na escala de anticorpos, ao se ajustar um limite para a concentração de anticorpos, os mesmos serão melhorados de uma forma seletiva.

Outro operador que será abordado neste trabalho, será o Operador de Recombinação por Transformação de Bactérias (RTGA), de acordo com (TASSIANO, 2017). Este operador é baseado no processo biológico de bactérias que realizam a mudança em seu DNA utilizando outros DNA's dispersos no ambiente em qual elas se encontram. Este DNA é proveniente de bactérias que estão mortas no ambiente, e não precisam ser da

mesma espécie da bactéria que irá absorver este DNA. O RTGA trabalha com uma população de indivíduos mortos, que é composta pelos indivíduos descartados da população do AG através da etapa de seleção de sobreviventes. É importante ser ressaltado que este operador não foi implementado neste trabalho, ele somente teve os seus resultados comparados ao final do trabalho, com os resultados que foram obtidos por (TASSIANO, 2017) em outra execução do *f*ramework, para fins comparativos de desempenho.

Após a adição de indivíduos na população de mortos, é realizado o cruzamento de cromossomos mortos com a população de indivíduos vivos. O cruzamento definido deve produzir apenas um único filho, este filho, será comparado com o seu pai vivo, e caso tenha um *f*itness melhor que o do pai, irá substituir o pai na população de vivos e irá incrementar em uma unidade o *f*itness de recombinação do pai que agora estará morto. Caso negativo, o filho gerado é descartado e uma unidade de *f*itness será decrementada de recombinação do pai morto. Após todos os cruzamentos permitidos terem sido realizados pela taxa de recombinação, acontecerá então uma seleção dos sobreviventes mortos, baseado no ranking de seu *f*itness de recombinação, aonde os indivíduos mortos com menor *f*itness serão eliminados.(TASSIANO, 2017)

1.2 Trabalhos Relacionados

Os autores (BALA; SHARMA, 2016) e (ELLISON; NANZER, 2017) fizeram uma análise de desempenho obtido pelo AG quando implementado com diferentes variações de *crossover*. Eles diferem entre si em qual variação de *crossover* gera os melhores resultados, único ponto de corte e uniforme respectivamente, mas o primeiro também conclui que uma baixa probabilidade de mutação favorece um melhor desempenho do AG. Outros autores (KHELIFA; BOUGHACI; AIMEUR, 2017) propuseram um novo operador de *crossover* que reproduziu melhores resultados do que os encontrados na literatura, através de comparações analíticas e qualitativas com funções de otimização, desvio padrão e tempo de execução.

Em termos de estudos comparativos de diversos operadores diferentes de AG, a comunidade acadêmica possui poucos estudos feitos, porém, alguns estudos estão na mesma linha de pesquisa deste trabalho, como por exemplo, a análise feita pelos autores (WU et al., 2014), tal estudo foi elaborado através de uma análise comparativa entre o AG, a Otimização de Enxame de Partículas (PSO) e o Rebanho de Krill.

Os três métodos estudados pelo autor foram utilizados para otimizar várias funções de desempenho, como por exemplo as funções Ackley's, Schaffer's e Rastrigin's que estão sendo utilizadas neste trabalho também. Após os testes o autor fez um análise dos resultados para descobrir a eficiência dos algoritmos. O autor utilizou métricas como tempo de execução, média e desvios padrões.(WU et al., 2014)

Recentemente, novos algoritmos meta-heurísticos foram propostos, mas os desempenho destes algoritmos na resolução de problemas de otimização, entretanto, estes algoritmos não são testados para diversas funções de otimização com outros operadores novos para ser feito um teste de desempenho. O autor (SAI et al., 2017) executou uma comparação de desempenho dos seguintes algoritmos, AG, PSO, Algoritmo do Lobo Cinza, Algoritmo *Brain Storm* e o Algoritmo das Libélulas. Para o seu teste de desempenho, o autor usou além das funções de Rastrigin's e Rosebrock's, outras funções para fazer comparação de precisão das melhores soluções encontradas, qualidade de soluções e tempo de convergência. Os resultados da comparação feita pelo autor se tornaram uma referência útil, para a escolha do algoritmo de otimização necessário para a resolução de problemas para a comunidade acadêmica.

Outro autor (LUTHRA et al., 2017) apresenta um estudo comparativo entre o Algoritmo de Morcegos, Algoritmo da Colônia de Abelha Artificial e Algoritmo da Colônia de Formigas. Algumas funções utilizadas para o seu teste também são utilizadas neste trabalho, após a execução o autor concluiu que o fator mais expressivo no teste foram os parâmetros utilizados para cada algoritmo estudado.

Atualmente, duas formas se tornaram as mais empregadas para manter a diversidade genética em um processo evolutivo. A primeira é gerenciar ativamente o processo, por exemplo, pelo controle de parâmetro dos operadores de variação. Caso a população comece a convergir para um ótimo local, é possível ajustar operadores de variação, ou seja, operadores de recombinação e mutação, ou a operação de seleção. Este método é conhecido principalmente como auto adaptação do Algoritmo Evolucionário (AE), mas requer uma sobrecarga adicional computacional e de implementação. A segunda maneira principal é manter passivamente a diversidade genética, como por exemplo, escolhendo modelos populacionais apropriados, como o autor (MANSO; CORREIA, 2015) faz no operador de parasitas. A ideia principal é dividir a população e operar em vários indivíduos com um certo nível de isolamento. No caso de um indivíduo dominar todos os outros, requer mais etapas para que esse indivíduo assuma a população inteira.(MUELLER-BADY et al., 2016)

Neste trabalho, também se buscou trabalhos acadêmicos que implementam novos operadores genéticos que modificam a estrutura do AG. Os operadores genéticos selecionados para análises foram o TGA (AMARAL; HRUSCHKA, 2014), SMuGA (MANSO; CORREIA, 2015), e IGAIE (LIU; YANG; PAN, 2015).

1.3 Referencial Teórico

Muitos problemas práticos no mundo real podem ser transformados em problemas de otimização multimodal[1], como otimização combinatória, design de módulo e planejamento de caminho. Esses problemas podem ser transformados em solução para função multimodal por meio de modelagem e simulação. Uma fraca capacidade de busca local e convergência prematura do algoritmo genético padrão é o efeito que é causado quando se é focado na resolução de problemas de otimização em funções multimodais.

O autor (JAVIDI; HOSSEINPOURFARD, 2015) explica que a diversidade do algoritmo genético do Caos evita a convergência local com mais frequência do que o AG tradicional. Além disso, resultados numéricos mostram que o método proposto diminui o número de iterações em problemas de otimização e melhora significativamente o desempenho do AG básico. A ideia de utilização de sequências caóticas para algoritmos de otimização é motivada por sistemas biológicos, como otimização de Enxame de Partículas, Algoritmo Colônia de Formigas e algoritmo de Colônia de Abelhas, e tem o potencial de melhorar AGs comuns.

De acordo com (MUELLER-BADY et al., 2016), uma abordagem relacionada ao modelo de difusão AEs, são AEs celulares, que também mantêm subpopulações espaciais dentro de uma população. A ideia subjacente a este método está intimamente relacionada com a dos autômatos celulares em outras áreas. Tendo uma população cujos indivíduos são distribuídos em uma grade, cada indivíduo mantém uma certa vizinhança, com a qual esse indivíduo é capaz de interagir, ou seja, recombinar ou substituir. Esse AE celular pode ser síncrono, quando todo o processo evolutivo é executado simultaneamente ou, de outra forma, assíncrono.

Um problema que surge nos AEs é a convergência prematura da população em espaços de busca multimodal. É frequentemente observado que o processo evolutivo converge cedo em um ótimo local arbitrário devido à deriva genética. A técnica do autor (MUELLER-BADY et al., 2016) neutraliza a convergência prematura, levando à melhoria contínua dos resultados. Ele age injetando um número controlado de indivíduos aleatórios na população e, assim, adicionando diversidade genética suficiente para manter o processo explorando novas áreas no espaço de busca.

Em (JAVIDI; HOSSEINPOURFARD, 2015) os autores apresentam um novo AG baseado em sistemas caóticos para superar a falha de convergência prematura. Nesse algoritmo, é empregado um mapa logístico e um mapa de tendas como dois sistemas caóticos para gerar valores caóticos em vez dos valores aleatórios nos processos AG. A diversidade do Algoritmo Genético do Caos evita a convergência local com mais frequência

[1] otimização multimodal significa procurar todas ou a maioria das melhores soluções, ou no mínimo o ótimo local de um problema, ao invés de procurar somente uma única solução

do que o GA tradicional. A ideia de utilização de sequências caóticas para algoritmos de otimização é motivada por sistemas biológicos.

Algumas funções de teste foram utilizadas também no trabalho do autor (KARGUPTA, 1996) no qual o mesmo relata o experimento para teste de desempenho do algoritmo genético bagunçado de expressão gênica, o mesmo enfatiza o papel da expressão gênica na evolução e procura por relações entre genes usando transcrição com os operadores, o autor utilizou as funções de Griewank's, Sheke's e Michalewicz's, os seus dados foram obtidos após um total de quinze execuções dos testes. As funções Rosenbrock's e Shekel's Foxholes foram utilizadas no trabalho feito pelo autor (FERNADEZ-VILLANCAS; AMIN, 1997) no qual o mesmo propôs um AG utilizando o histórico de busca de modo a reduzir o número de avaliações de *f*itness para aplicações de AG.

O Operador de Recombinação por Transformação (RTGA) que foi criado pelo autor (TASSIANO, 2017) é baseado no processo biológico de certas bactérias das quais realizam mudança em seu material genético através de outros materiais dispostos no seu meio. Este operador foi criado pelo seu autor ao utilizar o *Framework Evolutionary Algorithms* (FEA) que também foi utilizado neste trabalho. No capítulo 3 deste trabalho, esse *f*ramework tem o seu funcionamento explicado e as modificações que foram feitas para utilizarmos o mesmo neste trabalho.

O autor (TASSIANO, 2017), utilizou o framework incluindo o seu operador que foi criado recentemente junto com outros operadores, como o operador de meiose, o operador de bactérias, o operador imune e o operador do retrovírus , este último que serviu como base para o autor criar o operador RTGA.

1.4 Motivação

Um problema que surge nos AGs é a convergência prematura da população em espaços de busca multimodal. No caso desses cenários de pesquisa complexas, é frequentemente observado que o processo evolutivo converge cedo em um ótimo local arbitrário devido à deriva genética (MUELLER-BADY et al., 2016).

Existem várias propostas para tentar solucionar este problema, a mais comum é a de ajustes dos parâmetros do AG. Outras abordagens incluem novos operadores além dos operadores tradicionais de mutação e cruzamento. Esses aprimoramentos incluem, a mudança de gene para incluir genes mais relevantes através do operadores transgênico (AMARAL; HRUSCHKA, 2011), ou a divergência de material genético com o operador de parasita feito por (MANSO; CORREIA, 2015), ou no controle de processo de diversidade baseado na entropia da informação utilizado em (LIU; YANG; PAN, 2015).

Este trabalho tomou como motivação o cenário atual da academia com relação à

AGs no geral, aonde faltam estudos comparativos analisando o desempenho entre suas variantes e incrementos em forma de novos operadores. Além disso, contribuir com trabalho proposto por (TASSIANO, 2017), no qual é criado um framework para testes de novos operadores para AGs.

1.5 Justificativa

Os AGs tem sidos amplamente utilizados em problemas de otimização, o que torna essas técnicas alvo de constantes tentativas de melhorias, seja em novas formulações dos principais operadores, de mutação e cruzamento; seja na criação e adequação de novos operadores que forneçam alguma melhora de desempenho do algoritmo genético. Essas novas técnicas, quando propostas por seus autores, geralmente são comparadas com o algoritmo genético padrão para mostrar a melhoria de seu desempenho e cada autor define seus parâmetros de comparação.

Tendo em vista o crescimento dos números de novos operadores e variantes do AG convencional, faz-se necessário uma comparação justa entre esses novos operadores afim de mostrar qual deles apresenta os melhores resultados.

Sendo assim, comparar o desempenho destas técnicas é importante para fornecer uma base de conhecimento para a comunidade cientifica que mostre o comportamento desses algoritmos no que diz respeito a encontrar soluções e garantia de convergência.

Por tanto, este trabalho foi realizado com o objetivo de comparar outros algoritmos não inclusos ainda no Framework elaborado por (TASSIANO, 2018) e testar esses algoritmos de forma exaustiva mantendo a compatibilidade dos testes. Além disso, como o trabalho realizado por (TASSIANO, 2017) limita-se aos algoritmos de representação real, estendê-lo para a representação binária e comparar os algoritmos conforme seu tipo de representação. Com isso, espera-se que os resultados apresentados sirvam de parâmetro para a determinação de qual técnica é melhor empregada de acordo com a natureza do problema e também inclusão de novos operadores, o que torna a pesquisa mais relevante.

1.6 Objetivo Geral

Comparar novos operadores de AG, presentes no estado da arte, diferentes dos tradicionais (mutação, cruzamento) afim de fornecer informações sobre o comportamento destes para os usuários. Além disso, contribuir com o crescimento do Framework iniciado por (TASSIANO, 2017).

1.6.1 Objetivos Específicos

- Implementar os algoritmos presentes no estado da arte.
- Definir as melhores funções para avaliar o desempenho de AGs
- Categorizar os melhores desempenhos dos operadores utilizados
- Analisar os resultados dos operadores obtidos
- Comparar os resultados obtidos com os do Framework (TASSIANO, 2017)

1.7 Estrutura do Trabalho

Este trabalho está organizado em 5 (cinco) capítulos, listados a seguir: Introdução, Operadores e Funções de Teste Implementados, Resultados e Discussões e por último Considerações Finais.

1. **Introdução** : Visão geral do trabalho, incluindo a motivação, justificativa, trabalhos relacionados, referencial teórico e metodologia utilizada.
2. **Operadores e Funções de Teste Implementados** : Conceitos importantes para o entendimento do projeto, cada operador e função de teste que foi implementado para o projeto está neste capítulo.
3. **Processo de Desenvolvimento** : Visão geral do trabalho para o desenvolvimento dos operadores estudados neste trabalho.
4. **Resultados e Discussões** : Detalhes sobre os procedimento adotados nos testes, incluindo informações dos algoritmos, funções objetivos usadas, gráficos de desempenho e análises.
5. **Considerações Finais** : Considerações gerais sobre a pesquisa, incluindo as dificuldades encontradas, os pontos positivos e negativos da análise de desempenho.

2 OPERADORES E FUNÇÕES DE TESTE IMPLEMENTADOS

2.1 Operadores Implementados

2.1.1 Operador Transgênico

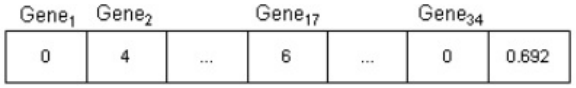
Muitas novas ideias da área genética foram incorporadas aos AGs atuais, tais como: cruzamento haploide, mutação, diploide, inversão, gene duplicado, deleção, dominância, translocação, diferenciação sexual e intrões. Mecanismos inspirados por eles poderiam ser potencialmente usados de maneira excelente na solução de problemas com AGs. Nos últimos anos, uma quantidade enorme de informação sobre AGs foi aprendido na comunidade genética sobre como os genes regulam uns aos outros - como eles ligam e desligam um ao outro de maneiras complexas, de modo que apenas os genes apropriados são expressos em uma dada situação. Isto é, essas redes regulatórias que tornam o genoma complexo, porém com um sistema extremamente adaptativo. Capturando esse tipo de genética, a adaptabilidade será cada vez mais importante, já que os AGs são usados em ambientes mais complicados e mutáveis. Seguindo essas ideias, o autor propôs um operador para algoritmos evolutivos inspirados em organismos transgênicos geneticamente modificados, especificamente com um mecanismo de transformação, onde características importantes são introduzidas em seus genomas artificialmente.(AMARAL; HRUSCHKA, 2011)

De acordo com o trabalho feito por (AMARAL; HRUSCHKA, 2011), o operador **Transgenic Genetic Algorithm** (TGA), é inspirado pela engenharia genética, onde foi manipulado o material genético dos indivíduos, adicionando características que se acreditam serem importantes. Nesse sentido, a abordagem pode ser vista como uma estratégia de elitismo focada em genes. Por conseguinte, o operador proposto identifica os genes com valores específicos e mantém esses valores nos genes alvo dos próximos indivíduos. Em outras palavras, se um gene G é identificado como tendo forte influência na função *f*itness quando se tem um valor específico V, então, o operador transgênico vai forçar alguns indivíduos na próxima geração a ter o valor V no gene G. O TGA foi aplicado em bases de dados cujos registros foram caracterizados por dados como idade, histórico familiar e uma série de sintomas.

O operador transgênico cria uma estrutura de histórico para guardar as informações que são consideradas relevantes (os genes) para o aumento da aptidão do indivíduo. Para isso, essa estrutura que tem tamanho definido por K, onde K é o tamanho da representação do indivíduo, guarda cada posição do gene do melhor indivíduo já encontrado

em todas as gerações. Junto dessa informação é guardado o valor de aptidão desse indivíduo e também o número de vezes que esse gene aparece em indivíduos com o mesmo valor do $fitness$, esse último que mostra a relevância do gene para o aumento do valor de aptidão. Cada vez que o aparece um indivíduo com o $fitness$ melhor que o do histórico na população, o histórico é atualizado com as informações do novo melhor indivíduo. A Figura 2 exemplifica como o esquema histórico é feito.

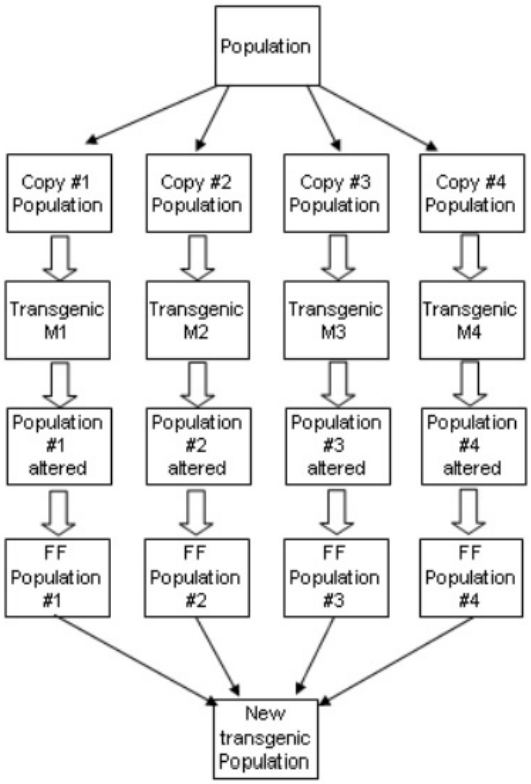
Figura 1 – Exemplo do Esquema Histórico.



Fonte – (AMARAL; HRUSCHKA, 2011)

Nesse operador são criadas 4 (quatro) cópias da população, depois disso, esses indivíduos são alterados por quatro módulos transgênicos M1, M2, M3 e M4 (AMARAL; HRUSCHKA, 2014). Os módulos transgênicos recebem como entrada uma cópia da população e uma lista com os genes que contribuem para obter o máximo de valor de aptidão da função. Esta lista é criada usando as informações sobre os genes presentes no histórico. Esses módulos juntos têm como saída uma única população alterada, esta população é avaliada e os N-indivíduos, onde N é o tamanho da população inicial, irão compor a nova população transgênica, estes passos estão ilustrados na Figura 3.

Figura 2 – Esquema dos Módulos do Operador Transgênico.



Fonte – (AMARAL; HRUSCHKA, 2011)

A escolha dos genes relevantes é feita probabilisticamente, utilizando uma roleta, isso implica que genes mais relevantes, que aparecem em mais indivíduos com valores de $fitness$ elevado, são mais propensos a serem escolhidos.

Cada módulo modifica apenas uma quantidade predefinida de genes nos indivíduos da população de entrada. O M1 muda somente um gene, por exemplo, se o gene 17 é sorteado pela roleta, o operador transgênico muda o valor desse gene para a mesma informação que está no histórico do gene de mesma posição. Do mesmo jeito que o módulo M1, os módulos M2, M3 e M4, também alteram os genes dos indivíduos, sendo que o M2 altera 2 genes, M3 altera 3 genes e M4, 4 genes. Os módulos trabalham em paralelo e no final, as quatro populações são compostas e o PSE é escolhido, onde PSE é o número de indivíduos da população. O TGA não é aplicado em todos os indivíduos da população, nesse trabalho somente 30(por cento) dos indivíduos são modificados.(AMARAL; HRUSCHKA, 2014)

2.1.2 Operador de Diversidade de Parasitas

Algoritmos evolutivos usando a simbiogênese foram propostos pela primeira vez por Nils Barricelli, o pioneiro dos algoritmos evolutivos. Enquanto o antigo concentra-se em estudar os efeitos da simbiogênese como um modelo de os fenômenos biológicos, este

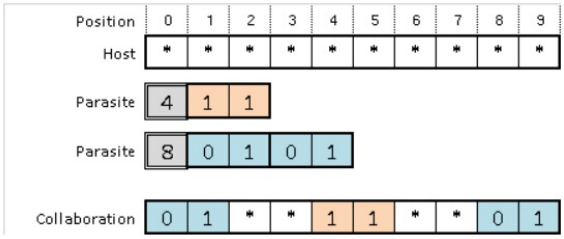
último aplica o modelo a simples problemas de otimização, função sinusoidal e redes neurais respectivamente. Em ambos os dois modelos, os organismos podem ser compostos por vários simbiontes. Estes podem ser de uma população, diferente na natureza da simbiose, como os neurônios formam uma rede neural, ou usando o conceito de hospedeiro e parasita, com o hospedeiro sendo uma composição recursiva de simbiontes, com algumas medidas de compatibilidade com a simbiose existente. (MANSO; CORREIA, 2015)

O **Symbiogenetic Multiset Genetic Algorithm** (SMuGA), utiliza um modelo de simbiogênese que foi baseado no Algoritmo Co-evolucionário Simbiogenético (SCA) (HEYWOOD; LICHODZIJEWSKI, 2010), em que duas populações, uma de hospedeiros e uma população menor de parasitas, combinam a cada interação. No SCA, apenas um parasita combina com um hospedeiro e o comprimento do genoma do parasita é constante. SMuGA é mais livre nesta regra, permitindo que múltiplos parasitas formem uma combinação com único hospedeiro, e variando o comprimento dos genomas do parasita na evolução.

O SMuGA é um algoritmo que utiliza duas espécies cooperativas, hospedeiros e parasitas, que evoluem juntos em um relacionamento mutualístico. A população de acolhimento contém soluções para o problema e a população de parasitas desenvolve soluções parciais. A população de parasitas evolui para alcançar bons genes e depois integrar os hospedeiros a incorporação desses genes, um passo chamado infecção por clareza, embora o efeito não seja necessariamente negativo para o hospedeiro. A interação entre hospedeiros e parasitas produz uma nova população usando simbiose que imita o que ocorre no ambiente natural mundo, além disso, colaboração é o resultado de um hospedeiro infectado por um ou mais parasitas, usando simbiose.

De acordo com o autor (HEYWOOD; LICHODZIJEWSKI, 2010), parasitas são compostos por uma tupla <posição, genoma>. A posição representa a localização no genoma hospedeiro onde o genoma do parasita começa a ser aplicado e o genoma representa o material genético do parasita. O genoma do hospedeiro é substituído pelo genoma do parasita no local definido pela posição atributo (Figura 3). O parasita considera o genoma do hospedeiro como um anel, o que significa que, ao aplicar o genoma do parasita ao hospedeiro, se chegarmos ao final do genoma do hospedeiro, a cópia continua no começo desse genoma. Na Figura 3, o parasita p1 é aplicado aos alelos 4 e 5 do genoma hospedeiro e o parasita p2 é aplicado no hospedeiro alelos do genoma 8, 9, 0 e 1. A colaboração neste caso é a combinação de genes hospedeiros e os genes introduzidos por parasitas p1 e p2.

Figura 3 – Colaboração formada pela simbiose de um hospedeiro e dois parasitas.



Fonte – (MANSO; CORREIA, 2015)

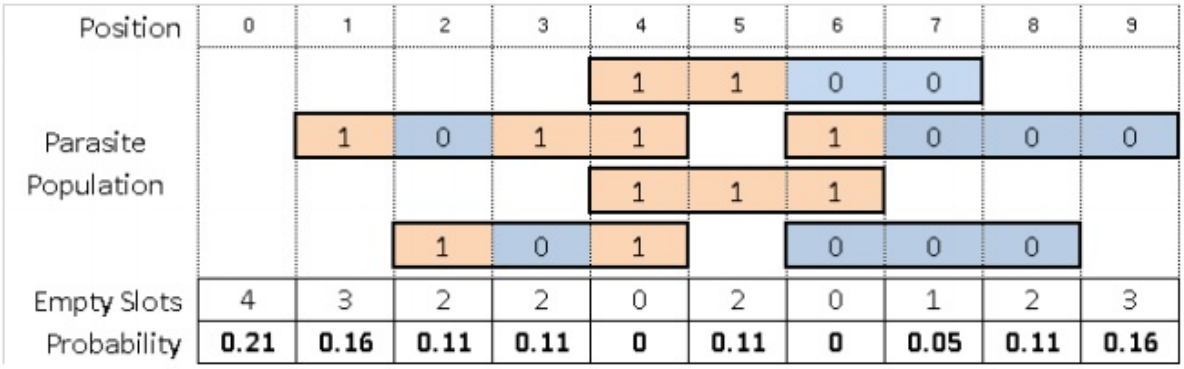
Para evitar que uma escolha humana interfira significativamente no desempenho, foi eliminada a necessidade de especificar o tamanho do parasita. Na abordagem que o autor utilizou (MANSO; CORREIA, 2015), o usuário não precisa saber o tamanho do problema dos *building blocks* (BB), caso existam, porque o algoritmo adapta o comprimento do genoma dos parasitas conforme necessário. Este sistema é importante para resolver problemas em que o tamanho do BB é desconhecido ou o BB tem um tamanho variável. O comprimento dos parasitas pode ser alterado por operadores genéticos de recombinação e mutação.

O sucesso do SMuGA é muito dependente da qualidade dos parasitas. Resultados anteriores (HEYWOOD; LICHODZIJEWSKI, 2010) mostraram que funções multimodais enganosas[1] eram difíceis de otimizar, sendo a principal razão atribuída ao baixa diversidade genética a população parasitária. Parasitas tendem a concentrar-se nos locais máximos e no fato de que a função tende a ser falsa, impede que eles alcancem novas regiões de espaço da pesquisa. Além disso, eles tendem a crescer em comprimento, sucessivamente propagando as soluções máximas locais e prevenindo ainda mais fuga e a possibilidade de alcançar o ótimo. A diversidade populacional do parasita rapidamente tende a valores baixos, para contornar esta dificuldade, a versão atual do SMuGA proposta por (MANSO; CORREIA, 2015) usa um conjunto de métricas para avaliar a cobertura do espaço do genoma. Estes métricas são usadas para ajudar os operadores genéticos a produzir e manter uma população parasitária diversificada ao longo da evolução.

A taxa de inoculação de cada gene hospedeiro é calculada dividindo o número de espaços vazios que o gene tem na população de parasitas poro número total de espaços vazios nessa população (Figura 4). Esta informação é usada na "posição de ancoramento da mudança" variante do operador de mutação, para desenhar probabilidades de iniciar um parasita em uma determinada posição do gene. Portanto, as mutações de posições parasitárias tendem a mover parasitas para regiões menos ocupadas do espaço do parasita. (MANSO; CORREIA, 2015)

[1] Uma função enganosa é uma função que tende a fazer com que o algoritmo acredite já ter encontrado o seu máximo local rapidamente e o mesmo pare com a sua pesquisa em novas regiões

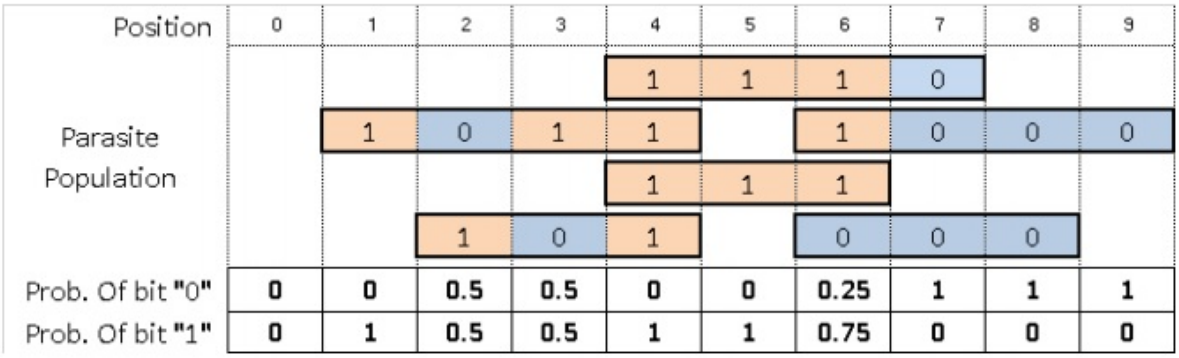
Figura 4 – Diversidade de ocupação de locus por parasitas.



Fonte – (MANSO; CORREIA, 2015)

Outra métrica computada sobre a população de parasitas é a diversidade de alelos em cada locus de hospedeiro. Para esse efeito calculamos as relações dos alelos 0 e 1 sobre o número total de parasitas. Em seguida, usamos essas proporções para modular o bit probabilidade de mutação de cada gene de acordo com o seu valor (Figura 5 probabilidade do *b*it "0" e Figura 5 probabilidade do *b*it "1"). Quanto maior a proporção maior a probabilidade de mudar o alelo, invertendo o *b*it valor. A fim de evitar que os parasitas se concentrem em pequenas regiões, o autor (MANSO; CORREIA, 2015) utilizou mais uma terceira métrica, modificando o *f*itness sendo que o mesmo vai ser levado em conta com a diversidade. Nesse primeiro caso, foi definida a diversidade de um parasita como a diversidade média de todos os seus alelos, com diversidade alélica definida como o complemento a taxa de alelo. Para cada parasita, seu valor de diversidade é usado para dimensionar a aptidão de avaliação descrita acima. O resultado final é que a aptidão de parasitas aglomerada em alguma região do espaço de pesquisa será rebaixada por comparação com parasitas isolados.

Figura 5 – Probabilidade de alelos parasitas em cada locus.



Fonte – (MANSO; CORREIA, 2015)

2.1.3 Operador Imune Adaptativo baseado na Entropia da Informação

(LIU; YANG; PAN, 2015) O **Immune Genetic Algorithm based on Information Entropy** (IGAIE), contém principalmente seleção de clonagem, distribuição de rede, seleção negativa, mecanismo de aprendizagem, mecanismo de memória, controle de concentração, vacinação e outras operações imunológicas. Estas ideias de algoritmo têm sido amplamente utilizadas em vários campos, como diagnóstico de falhas, segurança de rede, otimização de portfólio, detecção de anomalias e outros campos de engenharia. Os parâmetros do operador genético imunológico influenciam o efeito de otimização, no qual o ajuste adaptativo de probabilidade de cruzamento e probabilidade de mutação são maneiras de melhorar a convergência do algoritmo.

O operador imunológico é um operador de otimização global com um sistema de ajuste adaptativo com concentração de controle de anticorpos biológicos baseado no mecanismo imunológico biológico. De acordo com o autor (LIU; YANG; PAN, 2015), para manter a diversidade populacional de anticorpos, uma maior concentração de anticorpos leva à inibição, enquanto uma maior concentração de anticorpos leva à promoção. Alguns tópicos importantes para este operador são a diversidade e a concentração e o mecanismo de memória.

A entropia da informação é um indicador de medição semelhança entre anticorpos. Afinidade de anticorpos indica uma similaridade entre dois anticorpos, considerando a diversidade geral, usando a entropia da informação como o índice de similaridade reflete a similaridade mais compreensivamente do que usando a distância hamming. A concentração de anticorpos é utilizada para representar o número de escala de anticorpos. A concentração pode ser limitada para um alcance aonde o efeito Baldwin entre anticorpos no qual irá apropriamente impulsionar anticorpos melhores.(LIU; YANG; PAN, 2015)

A operação de seleção imunológica escolhe a melhor anticorpo no espaço imune que garante a convergência. A probabilidade de seleção do operador imunológico contém afinidade e concentração de anticorpos, que simula o mecanismo de regulação da concentração biológica do sistema imunológico. De acordo com o autor (LIU; YANG; PAN, 2015), a probabilidade de seleção do anticorpo é controlado pela afinidade e pela concentração, que é relação positiva com afinidade e relação inversa com concentração. Enquanto a concentração do anticorpo é certa, a probabilidade de seleção do anticorpo é positiva à afinidade e enquanto a afinidade do anticorpo é certa, a probabilidade de seleção do anticorpo é negativa para afinidade. Os anticorpos selecionados são conservados em células de memória, dessa forma, o anticorpo superior substitui o anticorpo com baixa afinidade em células de memória, de modo a atualizar a unidade de memória.

No operador imunológico, a probabilidade de cruzamento e probabilidade de mutação determina o desempenho do sistema imunológico algoritmo em grande medida. A

geração de novos anticorpos acelera tanto a probabilidade de *c*rossover e a probabilidade de mutação à crescer. Embora a diversidade de população de anticorpos é mantida, é fácil quebrar os genes bons originais. A probabilidade excessiva leva à busca aleatória, a probabilidade muito pequena não é propícia para gerar novos anticorpos, o que diminui a velocidade de pesquisa.(LIU; YANG; PAN, 2015)

Quando a diversidade de população de anticorpos é adequada, a operação de cruzamento e mutação deve ser reduzida. Enquanto quando a diversidade é inaptidão, o cruzamento e a mutação operação deve aumentar. De acordo com o autor (LIU; YANG; PAN, 2015), a interação entre o vários picos na operação de crossover devem ser evitados para proteger a direção de convergência. Uma maneira mais suave de evitar a interseção de valores ótimos multimodais sem configurar o limiar de distância constante é construída, isto é um processo adaptativo.

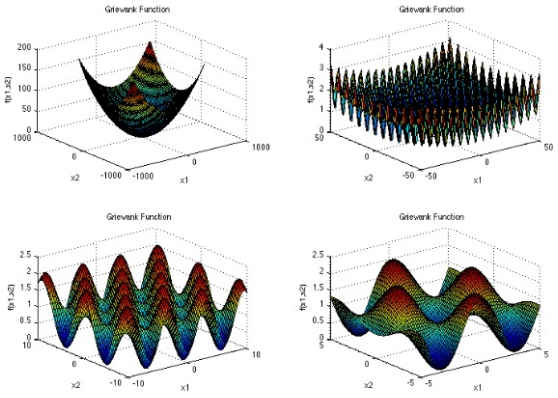
2.2 Funções de Teste

As funções de teste que foram utilizadas para teste de desempenho no experimento neste trabalho são as seguintes :

1. **Griewank's** : A função Griewank possui diversos mínimos locais difundidos, que são distribuídos regularmente. Esta função é amplamente usada para testar a otimização de convergência, a sua complexidade é mostrada nos gráficos ampliados. Logo abaixo segue a função matemática e a Figura 6 demonstra o gráfico da função. (SURJANOVIC; BINGHAM, 2018)

$$f(\mathbf{x}) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

Figura 6 – Gráfico da Função Griewank.

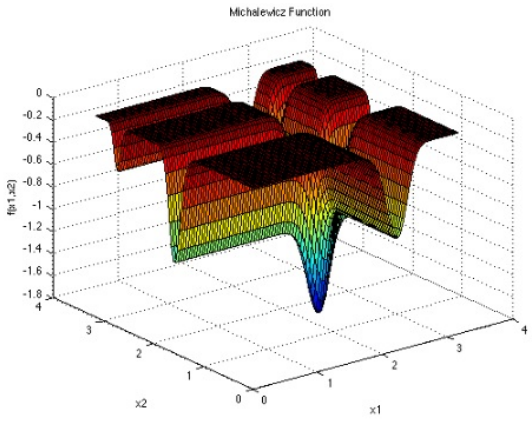


Fonte – (SURJANOVIC; BINGHAM, 2018)

2. **Michalewicz's** : A função Michalewicz tem d! mínimos locais, aonde d são as suas dimensões, e a função é multimodal. O parâmetro m define a inclinação de vales e cordilheiras; um m maior leva a uma busca mais difícil. A Figura 7 demonstra o gráfico da função. (SURJANOVIC; BINGHAM, 2018)

$$f(\mathbf{x}) = -\sum_{i=1}^{d} \sin(x_i) \sin^{2m}\left(\frac{ix_i^2}{\pi}\right)$$

Figura 7 – Gráfico da Função Michaelwicz.



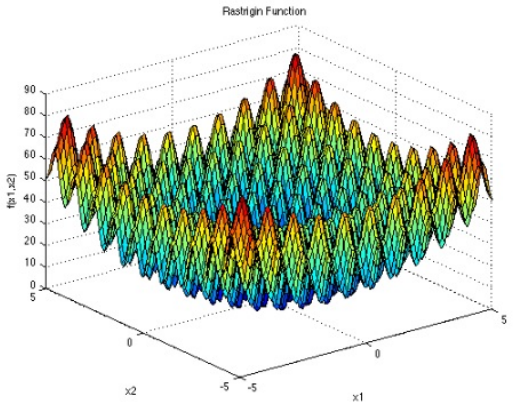
Fonte – (SURJANOVIC; BINGHAM, 2018)

3. **Rastrigin's** : A Função Rastrigin é frequentemente usada para testar algoritmos genéticos, porque seus muitos mínimos locais dificultam que o métodos padrão que

sejam baseados em gradientes encontrem o mínimo global, a mesma possui vários mínimos locais. A função é altamente multimodal, porem seus locais mínimos são proporcionalmente distribuídos. A Figura 8 demonstra o gráfico em sua forma bidimensional da função. (SURJANOVIC; BINGHAM, 2018)

$$f(\mathbf{x}) = 10d + \sum_{i=1}^{d} \left[x_i^2 - 10\cos(2\pi x_i)\right]$$

Figura 8 – Gráfico da Função Rastrigin.

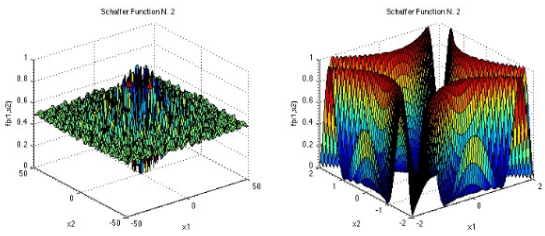


Fonte – (SURJANOVIC; BINGHAM, 2018)

4. **Schaffer's** : A segunda função de Schaffer, esta função é continua, unimodal e não-convexa, a mesma possui vários pontos de mínimos locais, porém a partir de sua borda até o seu centro acontece sempre uma minimização do valor da função o gráfico é demonstrada em sua forma dimensional de duas formas, uma em domínio amplo e outra em um domínio de escala menor para a visualização de detalhes. A Figura 9 demonstra o gráfico da função. (SURJANOVIC; BINGHAM, 2018)

$$f(\mathbf{x}) = 0.5 + \frac{\sin^2(x_1^2 - x_2^2) - 0.5}{\left[1 + 0.001(x_1^2 + x_2^2)\right]^2}$$

Figura 9 – Gráfico da Função Schaffer.

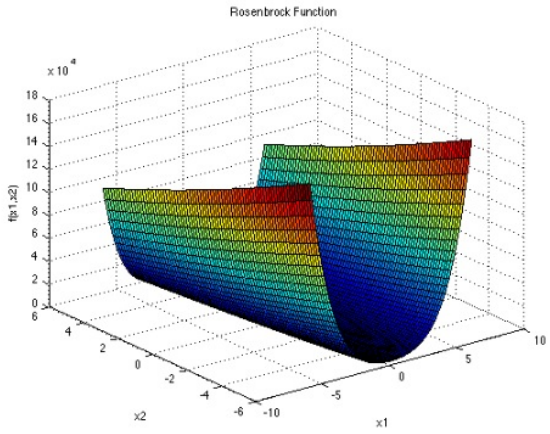


Fonte – (SURJANOVIC; BINGHAM, 2018)

5. **Rosenbrock's Valley** : A função Rosenbrock, também conhecida como a função Valley ou Banana, é um problema popular de teste para algoritmos de otimização baseados em gradiente. A função é unimodal e o mínimo global está em um vale estreito e parabólico. No entanto, embora esse vale seja fácil de encontrar, a convergência para o mínimo é difícil. A Figura 10 demonstra o gráfico em sua forma bidimensional da função. (SURJANOVIC; BINGHAM, 2018)

$$f(\mathbf{x}) = \sum_{i=1}^{d-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$$

Figura 10 – Gráfico da Função Rosenbrock Valley.



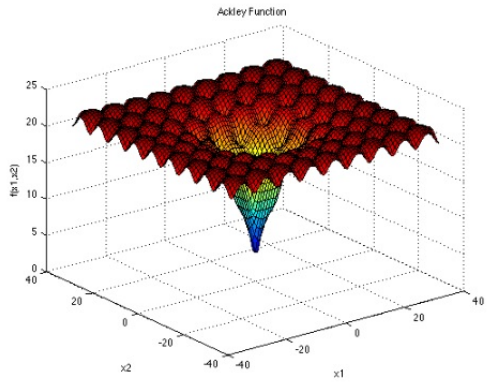
Fonte – (SURJANOVIC; BINGHAM, 2018)

6. **Ackley's** : A função Ackley é amplamente usada para testar algoritmos de otimização. Em sua forma bidimensional, como mostrado no gráfico abaixo, ela é caracterizada por uma região externa quase plana e um grande buraco no centro. A função representa um risco para os algoritmos de otimização ficarem presos em um de seus muitos mínimos locais, por conta de sua característica de possuir não

somente vários mínimos locais mais também que os valores entre esses mínimos locais são muito próximos do mínimo global e o ponto de convergência para seu mínimo global estar isolado de outros mínimos. A Figura 11 demonstra o gráfico da função.(SURJANOVIC; BINGHAM, 2018)

$$f(\mathbf{x}) = -a\exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + \exp(1)$$

Figura 11 – Gráfico da Função Ackley.



Fonte – (SURJANOVIC; BINGHAM, 2018)

7. **Shekel's Foxholes** : A função Shekel tem m mínimos locais. Abaixo estão os seus valores recomendados de m, o vetor B e a matriz C; B é um vetor m-dimensional e C é uma matriz 4-por-m-dimensional. A Figura 12 demonstra o gráfico em sua forma bidimensional da função. (SURJANOVIC; BINGHAM, 2018)

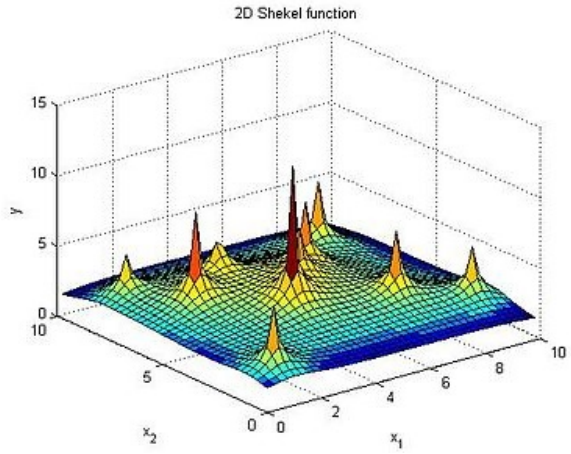
$$f(\mathbf{x}) = -\sum_{i=1}^{m}\left(\sum_{j=1}^{4}(x_j - C_{ji})^2 + \beta_i\right)^{-1}, \text{ where}$$

$$m = 10$$

$$\beta = \frac{1}{10}(1, 2, 2, 4, 4, 6, 3, 7, 5, 5)^T$$

$$\mathbf{C} = \begin{pmatrix} 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.6 \\ 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.6 \end{pmatrix}$$

Figura 12 – Gráfico da Função Shekel Foxholes.

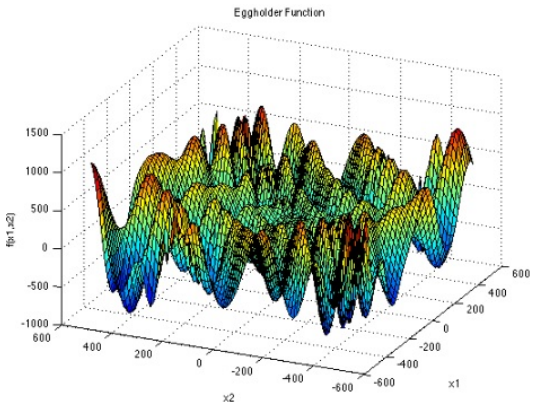


Fonte – (SURJANOVIC; BINGHAM, 2018)

8. **Eggholder** : A função Eggholder é extremamente difícil de otimizar devido à sua forma. Isso ocorre porque é caracterizado por um plano irregular que possui uma dezena de mínimos locais, com valores semelhantes ao seu único mínimo global. A Figura 13 demonstra o gráfico em sua forma bidimensional da função. (SURJANOVIC; BINGHAM, 2018)

$$f(\mathbf{x}) = -(x_2 + 47)\sin\left(\sqrt{\left|x_2 + \frac{x_1}{2} + 47\right|}\right) - x_1 \sin\left(\sqrt{|x_1 - (x_2 + 47)|}\right)$$

Figura 13 – Gráfico da Função Eggholder.



Fonte – (SURJANOVIC; BINGHAM, 2018)

Algumas funções foram utilizadas também no trabalho do autor (KARGUPTA, 1996) no qual o mesmo relata o experimento para teste de desempenho do algoritmo genético bagunçado de expressão gênica (GEMGA), o mesmo enfatiza o papel da expressão

gênica na evolução e procura por relações entre genes usando transcrição com os operadores, o autor utilizou as funções de Griewank's, Shekel's e Michalewicz's, os seus dados foram obtidos após um total de quinze execuções dos testes.

As funções Rosenbrock's saddle e Shekel's foxholes foram utilizadas no trabalho feito pelo autor (FERNADEZ-VILLANCAS; AMIN, 1997) no qual o mesmo propôs um AG utilizando o histórico de busca (MFEGA) de modo a reduzir o número de avaliações de $f$itness para aplicações de AG. O autor (YAW; CHONG; KAMIL, 2016) também usa em sua pesquisa as mesmas funções.

E as outras três funções restantes, o autor (ALI et al., 2010) também usou em seu trabalho para testar o desempenho do seu AG, no qual o mesmo é um híbrido entre dois métodos de otimização que são o sistema imunológico artificial (AIS) e o AG. O híbrido inclui dois processos; em primeiro lugar, o AIS é a atração entre os pesquisadores como o algoritmo. Isto permite-lhe desenvolver capacidade e eficiência de pesquisa local, embora a taxa de convergência para AIS não seja, de preferência, precisa em comparação com o AG. Em segundo lugar, um Algoritmo Genético está tipicamente inicializando a população aleatoriamente. A última geração de AIS será a entrada para o próximo processo do híbrido que é o AG neste híbrido AIS-AG. O híbrido faz com que o AG entre no estágio de soluções padrão mais rapidamente e com maior precisão em comparação com a população inicializada pelo AG aleatoriamente.

3 PROCESSO DE DESENVOLVIMENTO

Este trabalho trata-se de uma pesquisa exploratória na qual teve como objetivo a extração de informações dos operadores genéticos variantes de algoritmo genético, os quais tiveram uma abordagem de testes quantitativa e qualitativa com referências de procedimento bibliográfico.

Desta forma, o trabalho foi feito com a apresentação de três operadores variantes do AG, os quais são o operador transgênico (AMARAL; HRUSCHKA, 2011), o operador de parasitas (MANSO; CORREIA, 2015) e o operador imune adaptativo (LIU; YANG; PAN, 2015).

Alguns detalhes sobre o AG padrão utilizado são necessários serem especificados: A representação utilizada neste trabalho foi binária, para ser compatível com o que os autores dos operadores indicam em seus trabalhos. A seleção dos pais foi feita por *r*anking. O cruzamento utilizado foi o de um ponto de corte, e a mutação padrão *b*itflip. Entretanto, para implementação do SMuGA os autores apontam que devem ser utilizados 3 tipos de mutação, isso o diferencia do modelo de AG utilizado com os outros operadores.

Para testar os operadores implementados, foram utilizadas 8 funções multimodais, nas quais deve-se encontrar o mínimo global: Rastrigin's Function, Rosenbrock's Valley Function, Griewank's Function, Ackley's Function, Shekel's Foxholes, Michalewicz's Function e Schaffer Function. Essas funções são comumente utilizadas na literatura por possuírem um grau de complexidade alto para encontra seu mínimo global (KARGUPTA, 1996), (FERNADEZ-VILLANCAS; AMIN, 1997), (YAW; CHONG; KAMIL, 2016), (ALI et al., 2010). Essas funções possuem como característica muitos ótimos locais, o que torna a busca mais complexa, uma vez que o algoritmo pode ficar preso em uma dessas soluções sub ótimas. Dessa forma, utilizar as funções multimodais para testar os algoritmos objetiva mostrar o quão robusto é o algoritmo para fugir de ótimos locais. Além disso, verificar sua aplicabilidade em problemas reais, já que vários problemas no mundo real podem ser modelados matematicamente com funções multimodais.

Para cada um dos operadores foram feitas 30 (trinta) execuções independentes para cada função multimodal, para conferir validade estatística ao trabalho e avaliar a robustez em otimizar as funções. Como métricas para analisar a convergência da população, é calculada a aptidão do melhor individuo em cada geração. De posse dessas informações, é calculada a média e desvio padrão em cada geração das 30 (trinta) execuções do algoritmo. Ainda, é calculado, ao fim de cada execução o erro entre a melhor solução encontrada e solução real do problema, para medir o quanto próximo de encontrar a solução o algoritmo se aproximou. Dessa forma, pode-se medir a robustez para encontrar a solução do

problema. Neste trabalho, são consideradas 3 (três) precisões para indicar que a solução foi encontrada, quando o erro é menor ou igual a 10^{-3}, 10^{-2} e 10^{-1}.

Neste trabalho foram consideradas apenas funções de 2 (duas) dimensões para compatibilizar com os testes realizados por (TASSIANO, 2018) no FEA[1]. O *F*ramework utilizado nesse trabalho já contém as funções de teste que foram utilizadas, além de várias outras funcionalidades implementadas, como operador de mutação gaussiana, cruzamento aritmético, operadores para seleção dos pais e sobrevivente, entre outros. Ainda, esse framework fornece ao usuário, interfaces e classes abstratas que tornam a criação de novas rotinas mais facilitada. Contudo, como esse trabalho é recente, o autor apenas utilizou representação real no AGs, então foi necessário incluir a representação binária, e assim como seus operadores como, mutação *b*itflip e cruzamento por um ponto de corte, além de adaptação para incluir os operadores: transgênico (AMARAL; HRUSCHKA, 2011), o operador de parasitas (MANSO; CORREIA, 2015) e o operador imune adaptativo (LIU; YANG; PAN, 2015).

Durante a execução, foram testadas várias combinações de parâmetros para cada AG, sobre os quais os operadores são implementados. Na Tabela 1 encontram-se os parâmetros gerais do algoritmo utilizados para teste, neles incluem o número de gerações, que é o único critério de parada do algoritmo; o número de execuções do algoritmo para cada função de teste; o tamanho do *r*anking; o tamanho da população que foi utilizado o mesmo para todos os algoritmos; também o tamanho do indivíduos. As Tabelas 2, 3 e 4 mostram os parâmetros utilizados no SMuGA, TGA e IGAIE, respectivamente.

Tabela 1 – Parâmetros Gerais.

Parâmetros	
Tamanho da População	100
Tamanho dos Indivíduos	64
Tamanho do Ranking	3
Número de Gerações	1000
Número de Execuções	30

Fonte: O Autor.

[1] O diagrama UML de classes que mostra a organização do FEA encontra-se no Apêndice C.

Tabela 2 – Parâmetros Usados para o SMuGA.

SMuGA	
Número de Indivíduos Trocados	30
Taxa de Cruzamento	0.9
Taxa de Mutação	0.03

Fonte: O Autor.

Tabela 3 – Parâmetros Usados para o TGA.

TGA	
Número de Indivíduos Trocados	30
Taxa de Cruzamento	0.85
Taxa de Mutação	0.1
Taxa de Transgênico	0.3

Fonte: O Autor.

Tabela 4 – Parâmetros Usados para o IGAIE.

IGAIE	
Número de Indivíduos Trocados	50
α	0.9
β	1.1
λ	0.1

Fonte: O Autor.

4 RESULTADOS E DISCUSSÕES

Este capítulo apresenta os resultados obtidos, os gráficos de convergência das funções mais relevantes, o percentual de acertos dos algoritmos para cada uma das funções e o número de gerações necessárias para encontrar uma solução factível.

Também é mostrada a comparação entre os resultados obtidos por cada um dos operadores implementados, juntamente com os resultados obtidos por (TASSIANO, 2018) em experimento de mesma natureza comparativa com compatibilidade na metodologia utilizada.

Para tornar mais objetiva a apresentação dos resultados, foram selecionados apenas os gráficos de desvio padrão mais relevantes para análise. Para isso, foi escolhida a função Ackley e a função Eggholder, a primeira que mostra um comportamento característico do algoritmo[1] para as funções apresentadas e a última, que mostra um aumento do desvio padrão por se tratar de uma função mais complexa. É importante apresentar os resultados da função Eggholder, pois ela foi a função que apresentou menor percentual de acertos entre os algoritmos implementados neste trabalho.

4.1 Análise do Percentual de Acerto

Quando falamos em percentual de acertos do algoritmo, devemos entender que isso é relativo e depende da precisão desejada. Na análise do resultado de cada operador, é mostrado que os percentuais de acertos têm diferentes valores para cada precisão. Essa medida foi tomada para mostrar que por vezes o algoritmo não encontrou a resposta com maior precisão mas ficou muito perto. Isso pode ser visto no percentual de acertos das menores precisões. É importante compreender que uma técnica que consegue chegar à maiores precisões é mais relevante, pois os maiores valores de precisão englobam também os menores.

A Tabela 5 mostra o percentual de sucesso ao encontrar a resposta com uma precisão definida para cada uma das funções de teste.

Observa-se que o SMuGA se comportou melhor para as funções Ackley, Michalewicz, Rastrigin, Shaffer e ShekelFoxholes, uma vez que a taxa de sucesso para encontrar uma solução viável é maior ou igual a 87%, em todas essas funções. Quando comparamos a função Rastrigin, que obteve o menor sucesso entre as funções já citadas, com as funções Eggholder, Griewank e Rosenbrock, percebemos que percentual de acerto cai em 30% para Rosenbrock, 54% para Eggholder e 77% para Griewank, considerando a precisão 10^{-3}. O

[1] todos os gráficos de convergências das funções se encontram no Apêndice A.

Tabela 5 – Percentual de sucesso para encontrar solução

Função	**Precisão**	**Acerto (percentual)**		
		IGAIE	**SMUGA**	**TGA**
Ackley	10^{-3}	100%	100%	100%
	10^{-2}	100%	100%	100%
	10^{-1}	100%	100%	100%
Eggholder	10^{-3}	27%	33%	10%
	10^{-2}	27%	33%	10%
	10^{-1}	53%	67%	23%
Griewank	10^{-3}	40%	10%	27%
	10^{-2}	90%	37%	67%
	10^{-1}	100%	97%	100%
Michalewicz	10^{-3}	100%	100%	100%
	10^{-2}	100%	100%	100%
	10^{-1}	100%	100%	100%
Rastrigin	10^{-3}	100%	87%	100%
	10^{-2}	100%	87%	100%
	10^{-1}	100%	87%	100%
Rosenbrock	10^{-3}	100%	57%	100%
	10^{-2}	100%	67%	100%
	10^{-1}	100%	97%	100%
Shaffer	10^{-3}	100%	90%	100%
	10^{-2}	100%	100%	100%
	10^{-1}	100%	100%	100%
ShekelFoxholes	10^{-3}	87%	93%	27%
	10^{-2}	87%	93%	27%
	10^{-1}	87%	93%	27%

Fonte: O Autor.

que mostra um desempenho muito inferior do algoritmo para essas três funções. Mas se considerarmos a precisão 10^1, o percentual mínimo de acerto para todas as funções, antes 10%, sobe para 67%.

O TGA demonstrou um bom percentual de acertos para a maioria das funções. As que não obtiveram bons resultados foram as funções Eggholder, Griewank e ShekelFoxholes, sendo que o que mais chama atenção é a função Eggholder que teve pior desempenho se comparada as outras funções, pois apena 10% das vezes entrou o resultado com precisão 10^{-3}. Já as outras funções, o algoritmo conseguiu otimiza-las 100% das vezes.

O IGAIE demonstrou um bom percentual de acertos para a maioria das funções, sendo que as únicas que ficaram abaixo de 87% foram a Eggholder e a Griewank. Sendo que das 8 funções, 5 delas foram otimizadas 100% das vezes para todas as precisões utilizadas para teste.

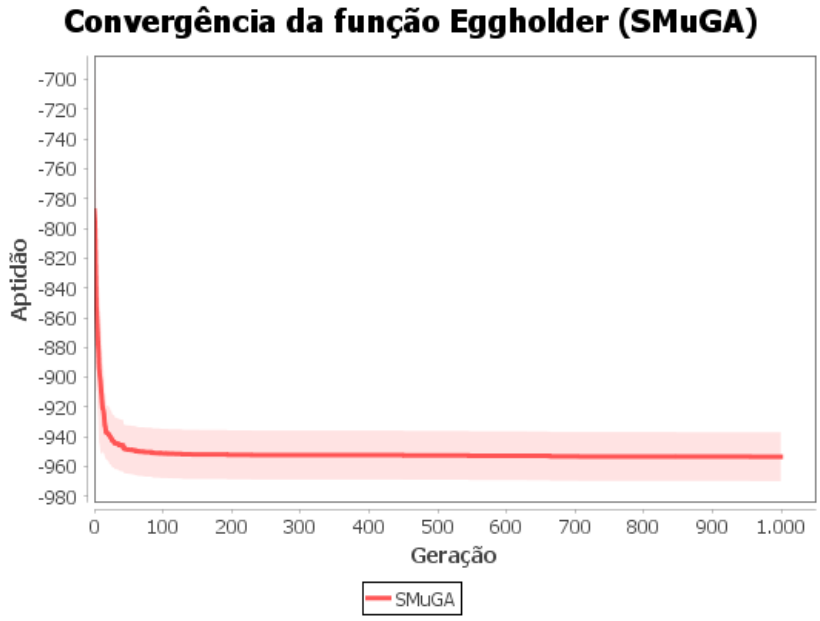
Nos testes realizados, pode-se notar que apenas 2 funções foram otimizadas com 100% de sucesso por todos os algoritmos, Ackley e Michalewicz. Já as funções que apresentaram menor taxa de acerto para todos os algoritmos, foram Griewank e Eggholder, considerando a precisão de 10^{-3}. Se considerarmos a precisão 10^{-1}, a única que apresentou um grau de complexidade maior para otimização, foi a Eggholder.

4.2 Análise do Comportamento do Algoritmos

Uma das métricas importantes na avaliação dos AGs é a análise de convergência, que mostra a capacidade do algoritmo de encontrar a solução do problema e fugir de ótimos locais. Neste trabalho, essa análise será feita de forma gráfica verificando o comportamento característico do algoritmo em várias execuções para cada função, como o definido na metodologia. Como o comportamento dos algoritmos foi bem parecido, na análise individual das duas funções, Ackley e Eggholder, só apresentaremos os gráficos do SMuGA [2].

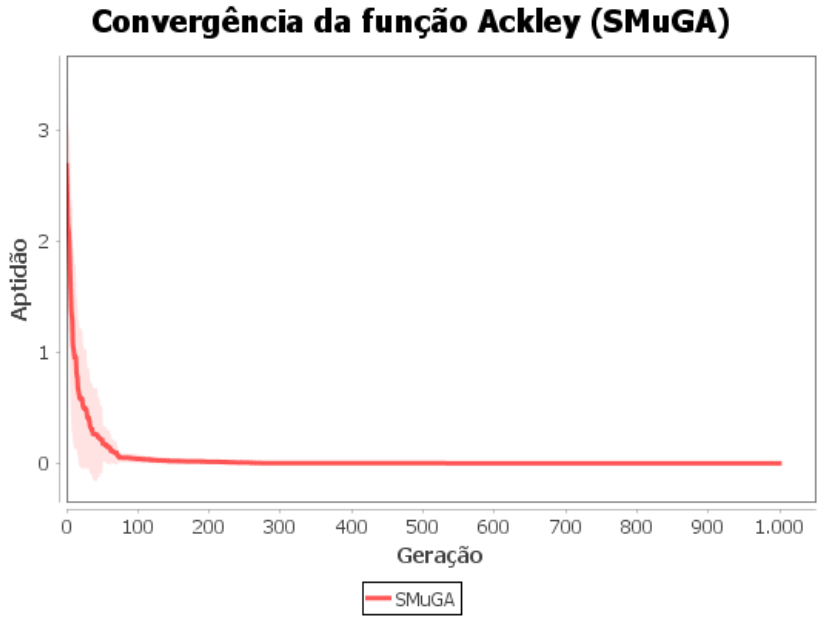
[2] Os gráficos de todas as funções podem ser encontrados no Apêndice A.

Figura 14 – Gráfico de Convergência da Função Eggholder (SMuGA).



Fonte: O Autor.

Figura 15 – Gráfico de Convergência da Função Rastrigin (SMuGA).

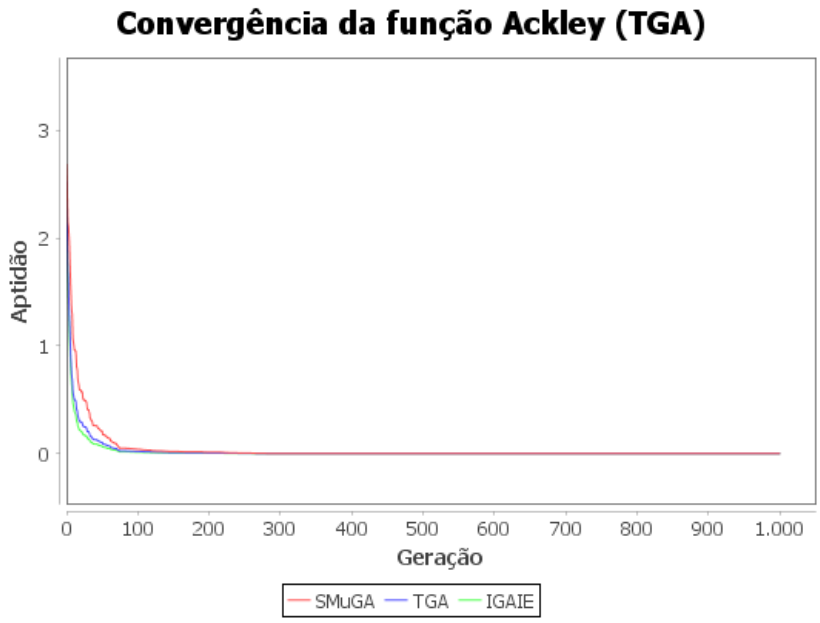


Fonte: O Autor.

A Figura 15 e Figura 14 mostram o comportamento dos algoritmos considerando o valor do $f$itness no decorrer das gerações. O gráfico apresenta a média do $f$itness do melhor indivíduo juntamente com o seu desvio padrão. Podemos observar na Figura 14 um valor alto de desvio padrão para o $f$itness médio no fim da execução do algoritmo (geração 1000). Isso indica que o algoritmo se comporta de forma bem diferente em cada uma das

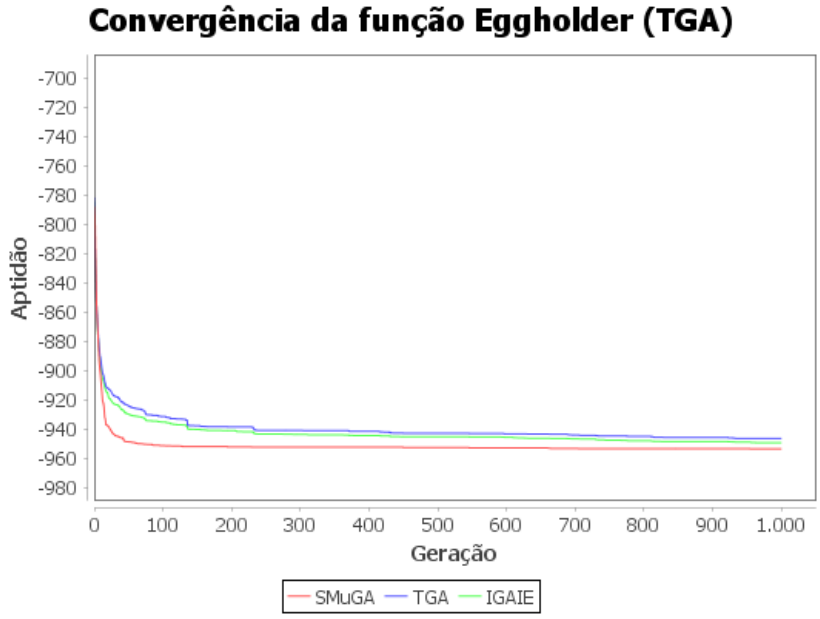
execuções. O mesmo não ocorre para o gráfico da função Ackley, que apresenta desvio padrão quase nulo no fim do processo, o que indica que o algoritmo tende a convergir para um ponto, nesse caso, o mínimo da função.

Figura 16 – Gráfico de Comparação da Função Ackley.



Fonte: O Autor.

Figura 17 – Gráfico de Comparação da Função Ackley.



Fonte: O Autor.

A Figura 16 e Figura 17 mostram a comparação das médias de *f*itness, entre os 3 (três) algoritmos, SMuGA, TGA e IGAIE para as funções Ackley e Eggholder. Para

a função Ackley, Figura 16, podemos verificar que quem teve melhor desempenho médio foi o IGAIE. Já para a função Eggholder, Figura 17, o algoritmo que apresentou melhor aproximação das soluções, foi o SMuGA. É interessante notar também que para a função Eggholder, o SMuGA teve o pior desempenho entre as técnicas. O TGA ficou entre as duas técnicas na função Eggholder e teve pior desempenho na função Ackley.

4.3 Análise da Velocidade de Convergência

A análise da velocidade convergência é importante para mostra o quão rápido o algoritmo convergiu para a uma solução factível. Nesse trabalho, fizemos essa análise por meio do número de gerações necessário para encontrar a solução, baseado em 3 valores de precisão definidos, 10^{-3}, 10^{-2} e 10^{-1}. A Figura 6 mostra a média e Desvio Padrão (DP). A esquerda, encontram-se as funções de teste e ao lado, os vários valores de precisão. Acima, temos os nomes dos algoritmos seguidos das médias e DPs relacionados. Para facilitar a análise, os melhores valores de média estão sublinhados e negrito. Quanto aos melhores DPs, eles estão apena em negrito para destacar.

Tabela 6 – Média de gerações para encontrar solução factível

Função	Precisão	IGAIE		SMUGA		TGA	
		Média	DP	Média	DP	Média	DP
Ackley	10^{-3}	63,3	17,8	284,4	99,4	**28,2**	**7,6**
	10^{-2}	28,4	9,3	159,5	65,4	**16,4**	**3,4**
	10^{-1}	12,9	4,9	57,2	31,8	**8,8**	**2,7**
Eggholder	10^{-3}	785,8	363,2	**726,7**	394,0	934,0	**222,8**
	10^{-2}	764,9	398,5	**712,9**	413,3	933,0	**226,1**
	10^{-1}	561,1	449,9	**437,8**	424,6	826,2	**338,3**
Griewank	10^{-3}	**744,2**	388,4	933,0	**211,2**	792,4	361,8
	10^{-2}	**258,9**	**337,7**	757,0	344,2	440,6	442,8
	10^{-1}	**17,8**	**5,9**	158,3	183,8	15,1	12,7
Michalewicz	10^{-3}	16,9	7,2	61,8	43,1	**8,7**	**4,9**
	10^{-2}	9,1	7,1	19,0	16,9	**4,8**	**3,8**
	10^{-1}	3,4	1,4	4,7	2,9	**2,1**	**1,1**
Rastrigin	10^{-3}	40,2	**12,2**	313,1	305,1	**29,8**	34,6
	10^{-2}	31,3	**12,3**	266,7	324,8	**25,9**	35,3
	10^{-1}	**22,4**	**11,7**	228,6	338,4	22,8	35,7
Rosenbrock	10^{-3}	**72,1**	**101,9**	583,3	403,7	111,6	131,1
	10^{-2}	**21,6**	**29,1**	462,5	437,4	50,4	68,9
	10^{-1}	**2,9**	**2,9**	55,1	185,5	8,5	20,1
Shaffer	10^{-3}	**43,1**	**20,5**	253,2	319,7	75,3	149,0
	10^{-2}	24,8	**11,0**	78,9	63,3	**19,1**	14,2
	10^{-1}	5,1	3,4	10,9	9,3	**4,1**	**2,8**
ShekelFoxholes	10^{-3}	**149,8**	339,3	158,3	**256,4**	735,7	445,7
	10^{-2}	147,6	340,1	**145,7**	**258,9**	735,0	447,0
	10^{-1}	145,7	340,9	**135,3**	**260,5**	734,9	447,1

Fonte: O Autor.

O que podemos observar na Tabela 6 é que nenhuma das técnicas se sobressaiu sobre as outras em todas as funções. O TGA e IGAIE, para as funções: Ackley, Michalewicz, Rastrigin e Shaffer, apresentaram soluções factíveis antes da geração 80, em média, para todas as precisões. O SMuGA demora acima de 150 geração para devolver uma solução

viável, excetuando para a função Michalewicz, que ele obteve a resposta em média na geração 61,8, isso considerando a precisão de 10^{-3}. Logo abaixo a Tabela 7 traz os dados da Tabela 6 de forma mais limpa para uma melhor leitura dos dados mais importantes obtidos.

Tabela 7 – Média de gerações da Tabela 6 com as informações mais importantes.

Algoritmo	Média Gerações		
	IGAIE	SMUGA	TGA
Ackley	63	284	28
Eggholder	197	180	*340*
Griewank	*361*	*330*	221
Michelawicz	**17**	**62**	**9**
Rastrigin	40	207	30
Rosenbrock	72	265	112
Shaffer	43	170	75
ShekelFoxholes	19	98	9
Média	**102**	**200**	**103**
Desvio	**112**	**86**	**112**

Fonte: O Autor.

As Tabelas e mostram os resultados que foram obtidos e os resultados que eram esperados para cada função para cada operador executado, aonde os dois operadores que acertaram o valor de seis funções cada em seis funções diferentes, foram os operadores IGAIE e TGA.

Tabela 8 – Resultados esperados e os alcançados pelos operadores.

	Rastrigin	Michalewicz	Rosenbrock	Griewank
Esperado	0	-1,8013	0	0
IGAIE	**0**	-1,80130341	**6,75784E-14**	**0**
TGA	**0**	-1,80130341	1,15025E-12	**0**
SMuGA	1,77636E-15	-1,80130341	1,58928E-06	1,67955E-12
RIGA	7,879	**-1,801302849**	0,009251562	0,00761542
RTGA	**0**	-1,798451578	0,031923809	**0**

Fonte: O Autor.

Tabela 9 – Resultados esperados e os alcançados pelos operadores.

	Shekel	Schaffer	Ackley	EggHolder
Esperado	0,998003	0	0	-959,6407
IGAIE	**0,998003838**	**0**	4,31584E-08	**-959,6406619**
TGA	**0,998003838**	**0**	**0**	**-959,6406619**
SMuGA	**0,998003838**	4,44089E-16	6,58545E-09	**-959,6406619**
RIGA	0,998004	0,000417	4,84E-05	-932,795
RTGA	4,453621	**0**	**0**	-818,814

Fonte: O Autor.

4.4 Qualidade das Soluções

A qualidade das soluções foi analisada considerando o quanto próximo os algoritmos chegaram da solução real do problema. A Tabela 10 apresenta os resultados de melhor valor, média dos valores e pior valor. São utilizados três tons de cinza na coloração da tabela para tornar a visualização mais fácil. O cinza mais claro é o que representa valores que estão com mais de uma unidade de diferença do ótimo global, o cinza médio mostra valores que estão com uma diferença maior que 3 casas decimais e menor que uma unidade, e o cinza escuro mostra valores com precisão de 3 casas decimais ou mais. O operadores estão ordenado do melhor para o pior, foram considerados apenas os 5 melhores resultados, os outros operadores implementados em (TASSIANO, 2018) foram omitidos.

Tabela 10 – Comparação entre os Algoritmos

Operadores	Função	Rastrigin	Michalewicz	Rosenbrock	Griewank	Shekel	Schaffer	Ackley	EggHolder
	Resultado Esperado	0	-1,8013	0	0	0,998003	0	0	-959,6407
IGAIE	Best	0	-1,80130341	6,75784E-14	0	0,998004	0	4,31584E-08	-959,6406619
	AVG	2,45374E-13	-1,801303365	0,000286818	0,001233	1,032047	1,19164E-15	6,64723E-07	-954,6603774
	WORST	1,112E-12	-1,801303217	0,00074251	0,007397	1,406424	9,32587E-15	1,83107E-06	-893,701392
TGA	Best	0	-1,80130341	1,15025E-12	0	0,998004	0	0	-959,6406619
	AVG	2,54611E-15	-1,8013033	0,000201231	0,003945	1,18184	4,73695E-16	3,54717E-08	-939,1699876
	WORST	1,95399E-14	-1,801303217	0,00074251	0,007397	1,406423	1,42109E-14	8,63168E-08	-893,7011577
SMuGA	Best	1,77636E-15	-1,80130341	1,58928E-06	1,68E-12	0,998004	4,44089E-16	6,58545E-09	-959,6406619
	AVG	0,14888322	-1,801066688	0,015296612	0,025994	1,011642	0,000429556	1,42318E-06	-953,2704352
	WORST	1,235768941	-1,800711727	0,175808979	0,177693	1,202297	0,004863556	2,86801E-05	-893,7011471
RIGA	Best	7,879	-1,801302849	0,009251562	0,007615	0,998004	0,000417	4,84E-05	-932,795
	AVG	0,00534133	-1,80090468	0,012105888	0,004963	0,998004	0,000529	0,00121116	-925,287
	WORST	0,53335299	-1,761485968	0,293992464	0,005177	0,998004	0,00053	0,146327755	-925,06
RTGA	Best	0	-1,798451578	0,031923809	0	4,453621	0	0	-818,814
	AVG	0,00450612	-1,798371528	0,034437745	6,2932	6,828968	6,70E+07	0,000752511	-760,715
	WORST	0,45061163	-1,792447275	0,211399386	0,000629	39,12318	6,70E-05	0,075251067	-721,452

Fonte: O Autor - Adaptado de (TASSIANO, 2018)

A Tabela 10 mostra que os operadores SMuGA, TGA e IGAIE obtiveram os melhores resultados para encontrar o mínimo das funções Multimodais quando comparados com os outros algoritmos testados em (TASSIANO, 2018). Os melhores resultados obtidos foram do IGAIE, que encontrou o mínimo de todas as funções ao menos uma vez

durante os testes, inclusive nos testes com a Eggholder, que por sua vez foi a função que apresentou maior dificuldade para todos os algoritmos. Embora o IGAIE tenha mostrado resultados superiores, o TGA também mostrou excelente desempenho, mesmo que não tenha encontrado muitas vezes a solução para a Eggholder. É importante ressaltar que o operador transgênico apresentou boa convergência durante as execuções dos testes para quase todas as funções exceto para a Eggholder. O SMuGA obteve boas soluções para as funções testadas, tendo um bom desempenho no geral, mas não superando o TGA.

No contexto geral, é notável que o IGAIE apresentou melhores resultados para todas as funções de teste, e ainda quando comparamos com os resultados apresentados em (TASSIANO, 2018), podemos constatar que ele também obteve os melhores resultados. Além disso, as médias das soluções encontradas pelos algoritmos desenvolvidos neste trabalho foram mais próximas das soluções reais, o que mostra que no geral as soluções encontradas pelo SMUGA, TGA e IGAIE são melhores.

5 CONSIDERAÇÕES FINAIS

Neste trabalho foi realizada uma pesquisa bibliográfica em busca de novos operadores para AGs. Após isso foram escolhidos três operadores, SMuGA, TGA e IGAIE. Todos eles foram implementados conforme suas referências na literatura. Depois de implementados, os algoritmos passaram por teste de estresse utilizando-se de funções multimodais complexas de serem resolvidas.

Os resultados obtidos mostraram que todos os operadores implementados obtiveram bons resultados para otimizar as funções que foram utilizadas nos testes. O operador que mostrou melhor resultados considerando o percentual de acertos foi o IGAIE.

Os algoritmos demonstraram um comportamento parecido para as funções mostradas, Ackley e Eggholder, na primeira o desvio padrão era muito pequeno no fim do processo. Já na segunda, o desvio era maior, por nem sempre serem encontradas soluções de boa qualidade.

Quanto a qualidade das soluções, quem apresentou soluções mais próximas, foi o IGAIE, que perdeu apenas para o TGA na função Ackley.

Neste trabalho foi feita uma análise qualitativa e quantitativa de diversos operadores novos ou operadores que foram melhorados com 8 funções de teste já utilizadas na literatura, criando um banco de resultados com a análise desses novos operadores.

É interessante este trabalho ter uma extensão para uma expansão do banco de dados de resultados, e desta forma, as análises se tornam cada vez mais verossímeis com uma margem de erro menor a cada expansão do banco.

Como este trabalho limitou-se a manter os parâmetros dos AGs constantes nos testes de cada função avaliada. É importante fazer outra análise buscando a melhor configuração de parâmetros para cada função para comparar os algoritmos com seus melhores desempenhos.

Além disso, poderia ser feita uma expansão de análise comparativa, através das mesmas condições e parâmetros usados neste trabalho à outros operadores genéticos variantes de AG que não foram utilizados neste trabalho.

Outro estudo que pretende-se fazer, é comparar as funções de teste com diferentes dimensões e precisão melhor. Ainda fazer extensões deste trabalho comparando esses operadores com outras técnicas de otimização diferentes de AGs.

O código criado durante o trabalho pode ser encontrado para download no seguinte link,

Link do repositório : <https://bitbucket.org/adrianosbarreto/tcc/src/master/>

REFERÊNCIAS

ALI, M.; KOH, S.; CHONG, K.; YAP, D. Hybrid Artificial Immune System-Genetic Algorithm optimization based on mathematical test functions. *Research and Development (SCOReD), 2010 IEEE Student Conference on*, IEEE, n. SCOReD, p. 13–14, dec 2010. Disponível em: <http://ieeexplore.ieee.org/document/5704012/>.

AMARAL, L.; HRUSCHKA, E. Transgenic: An evolutionary algorithm operator. *Neurocomputing*, v. 127, p. 104–113, 2014. ISSN 09252312.

AMARAL, L. R. D.; HRUSCHKA, E. R. Transgenic, an operator for evolutionary algorithms. In: *2011 IEEE Congress of Evolutionary Computation (CEC)*. IEEE, 2011. p. 1308–1314. ISBN 978-1-4244-7834-7. Disponível em: <http://ieeexplore.ieee.org/document/5949767/>.

BALA, A.; SHARMA, A. K. A comparative study of modified crossover operators. *Proceedings of 2015 3rd International Conference on Image Information Processing, ICIIP 2015*, p. 281–284, 2016. ISSN 0149-2063.

EIBEN, A.; SMITH, J. *Introduction To Evolutionary Computing*. [S.l.: s.n.], 2003. v. 45.

ELLISON, S. M.; NANZER, J. A. Comparison of crossover recombination operators in GA-optimized sparse linear array design. In: *2017 IEEE International Symposium on Antennas and Propagation & USNC/URSI National Radio Science Meeting*. IEEE, 2017. p. 145–146. ISBN 978-1-5386-3284-0. Disponível em: <http://ieeexplore.ieee.org/document/8072115/>.

FERNADEZ-VILLANCAS, J.; AMIN, S. Simulated jumping in genetic algorithms for a set of test functions. *Proceedings Intelligent Information Systems. IIS'97*, IEEE Comput. Soc, p. 209–215, 1997. Disponível em: <http://ieeexplore.ieee.org/document/645223/>.

HEYWOOD, M. I.; LICHODZIJEWSKI, P. Symbiogenesis as a mechanism for building complex adaptive systems: A review. In: *Proceedings of the 2010 International Conference on Applications of Evolutionary Computation - Volume Part I*. Berlin, Heidelberg: Springer-Verlag, 2010. (EvoApplicatons'10), p. 51–60. ISBN 3-642-12238-8, 978-3-642-12238-5. Disponível em: <http://dx.doi.org/10.1007/978-3-642-12239-2_6>.

JAVIDI, M.; HOSSEINPOURFARD, R. Chaos genetic algorithm instead genetic algorithm. *International Arab Journal of Information Technology*, v. 12, n. 2, p. 163–168, 2015. ISSN 16833198. Disponível em: <http://ccis2k.org/iajit/PDF/vol.12,no.2/6039.pdf>.

KARGUPTA, H. The performance of the gene expression messy genetic algorithm on\nreal test functions. *Proceedings of IEEE International Conference on Evolutionary Computation*, IEEE, p. 631–636, 1996. Disponível em: <http://ieeexplore.ieee.org/document/542674/>.

KHELIFA, M.; BOUGHACI, D.; AIMEUR, E. An enhanced genetic algorithm with a new crossover operator for the traveling tournament problem. In: *2017*

4th International Conference on Control, Decision and Information Technologies (CoDIT). IEEE, 2017. p. 1072–1077. ISBN 978-1-5090-6465-6. Disponível em: <http://ieeexplore.ieee.org/document/8102741/>.

LIU, X.; YANG, Z.; PAN, W. An Improved Adaptive Immune Genetic Algorithm Based on Information Entropy. In: *2015 International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration*. IEEE, 2015. p. 6–9. ISBN 978-1-4673-8312-7. Disponível em: <http://ieeexplore.ieee.org/document/7373777/>.

LUTHRA, I.; CHATURVEDI, S.; UPADHYAY, D.; GUPTA, R. Comparative study on nature inspired algorithms for optimization problem. *Proceedings of the International Conference on Electronics, Communication and Aerospace Technology, ICECA 2017*, v. 2017-Janua, p. 143–147, 2017.

MANSO, A. M.; CORREIA, L. M. Parasite Diversity in Symbiogenetic Multiset Genetic Algorithm. In: *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference - GECCO '15*. New York, New York, USA: ACM Press, 2015. p. 887–894. ISBN 9781450334723. Disponível em: <http://dl.acm.org/citation.cfm?doid=2739480.2754780>.

MUELLER-BADY, R.; KAPPES, M.; BULO, I. M.; PALOMO-LOZONO, F. Leveraging diversity in evolutionary algorithms using a population injection method. In: *2016 20th International Conference on System Theory, Control and Computing (ICSTCC)*. IEEE, 2016. p. 526–531. ISBN 978-1-5090-2720-0. Disponível em: <http://ieeexplore.ieee.org/document/7790719/>.

SAI, V. O.; SHIEH, C. S.; LIN, Y. C.; HORNG, M. F.; NGUYEN, T. T.; LE, Q. D.; JIANG, J. Y. Comparative Study on Recent Development of Heuristic Optimization Methods. *Proceedings - 2016 3rd International Conference on Computing Measurement Control and Sensor Network, CMCSN 2016*, p. 68–71, 2017.

SURJANOVIC, S.; BINGHAM, D. *Virtual Library of Simulation Experiments: Test Functions and Datasets*. 2018. Disponível em: <http://www.sfu.ca/~ssurjano>. Acesso em: 23 de dezembro de 2018.

TASSIANO, G. Novo operador para algoritmos geneticos baseado na recombinação por transformação de bacterias: Um estudo comparativo em otimização multimodal. *Trabalho de conclusão de curso*, p. 74, 01 2017.

TASSIANO, G. Novo Operador Para Algoritmos Genéticos Baseado na Recombinação Por Transformação De Bactérias : Um estudo comparativo em otimização multimodal. *Projeto de Dissertação de Pós-Graduação*, p. 34, 01 2018.

WU, Q.; LI, C.; LI, Y.; SUN, H.; GUO, Q.; WU, J. SII and the fronto-parietal areas are involved in visually cued tactile top-down spatial attention: A functional MRI study. *NeuroReport*, v. 25, n. 6, p. 415–421, 2014. ISSN 1473558X.

YAW, M.; CHONG, K. H.; KAMIL, K. Transform of Artificial Immune System algorithm optimization based on mathematical test function. In: *2016 6th IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*. IEEE, 2016. p. 147–150. ISBN 978-1-5090-1178-0. Disponível em: <http://ieeexplore.ieee.org/document/7893561/>.

Apêndices

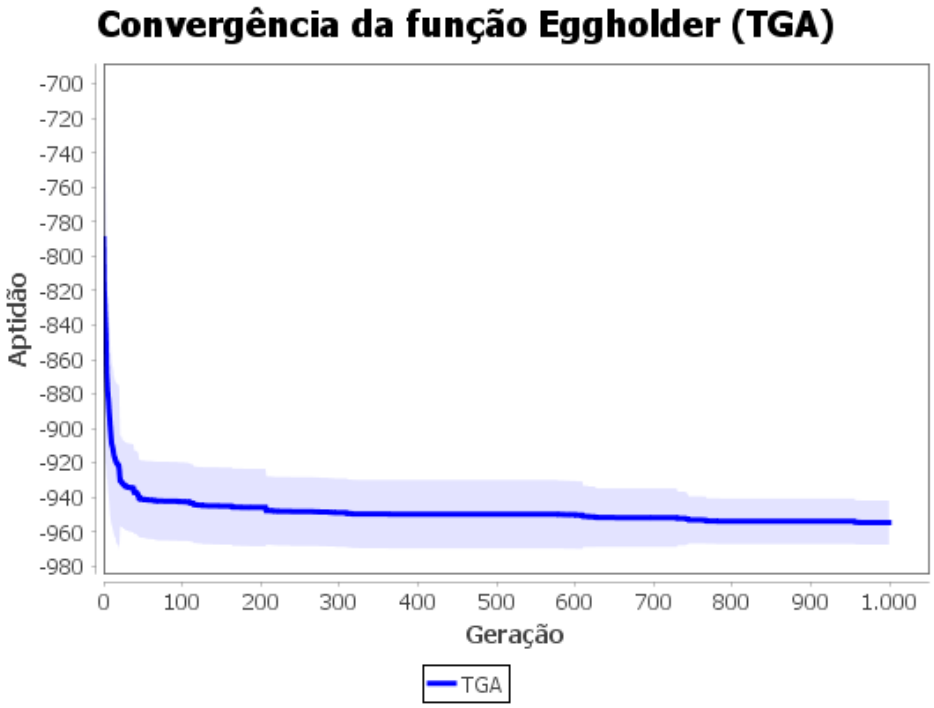
APÊNDICE A – GRÁFICOS DE CONVERGÊNCIA

A.1 Gráficos de Convergência para o Operador TGA

Figura 18 – Gráfico de Convergencia TGA da Função Ackley.

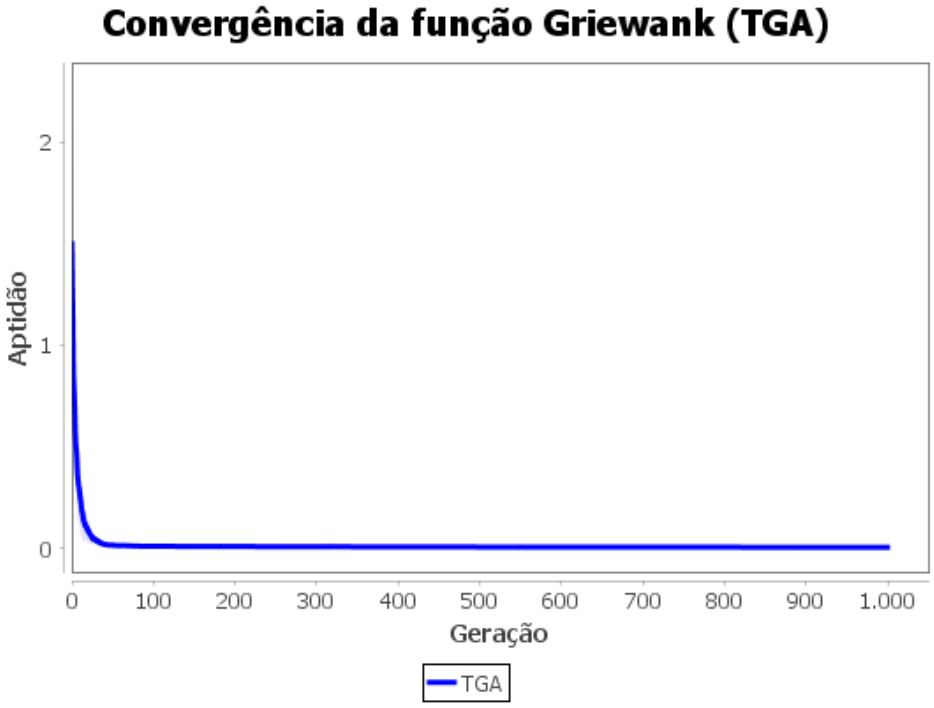
Fonte: O Autor

Figura 19 – Gráfico de Convergencia TGA da Função Eggholder.



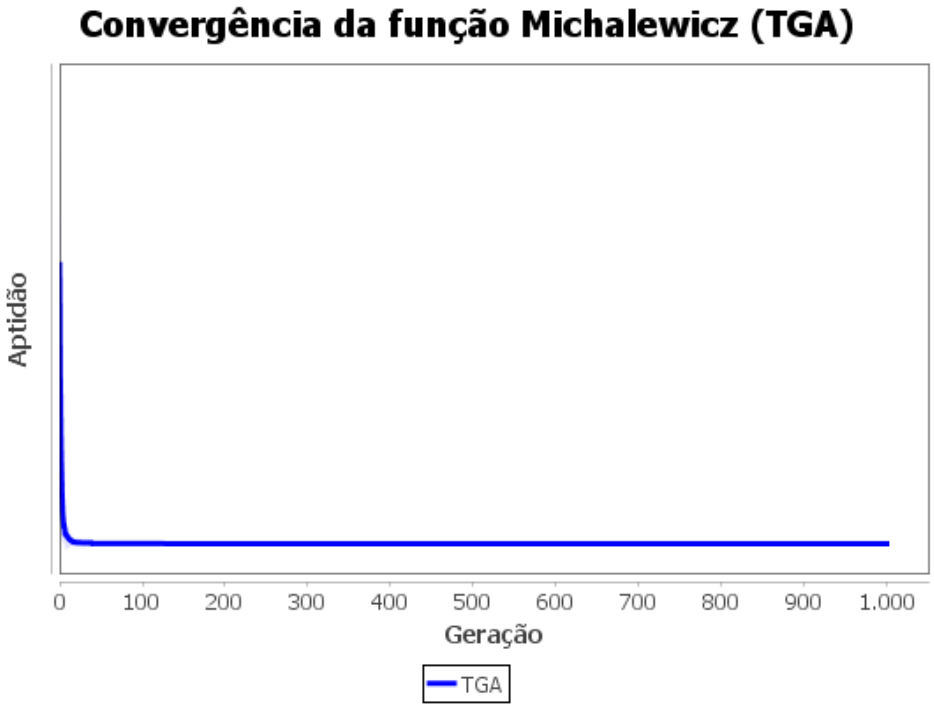
Fonte: O Autor

Figura 20 – Gráfico de Convergencia TGA da Função Griewank.



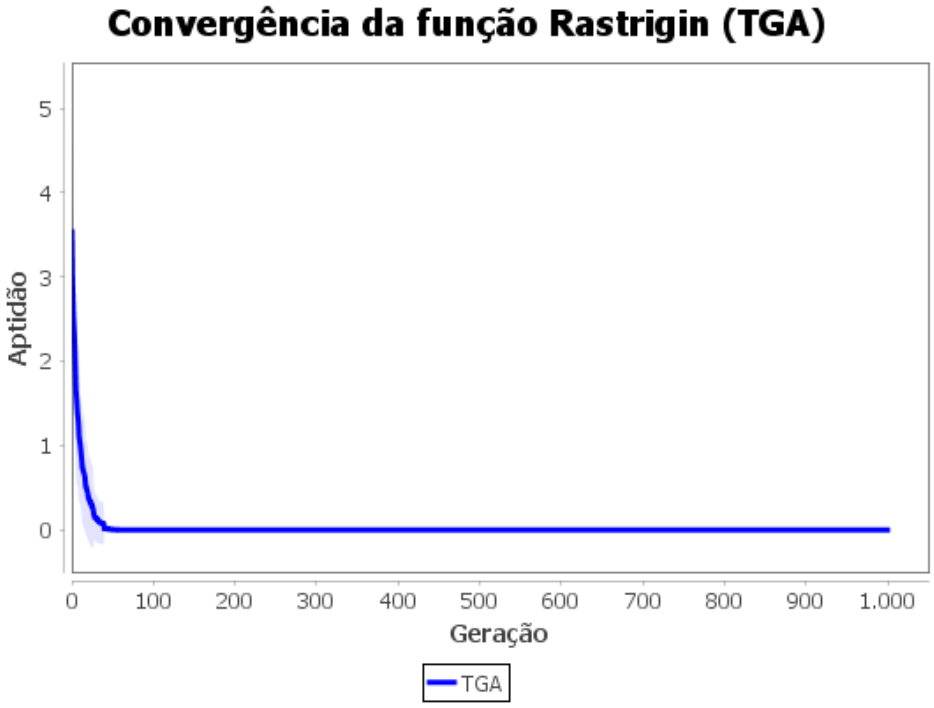
Fonte: O Autor

Figura 21 – Gráfico de Convergencia TGA da Função Michaelwicz.



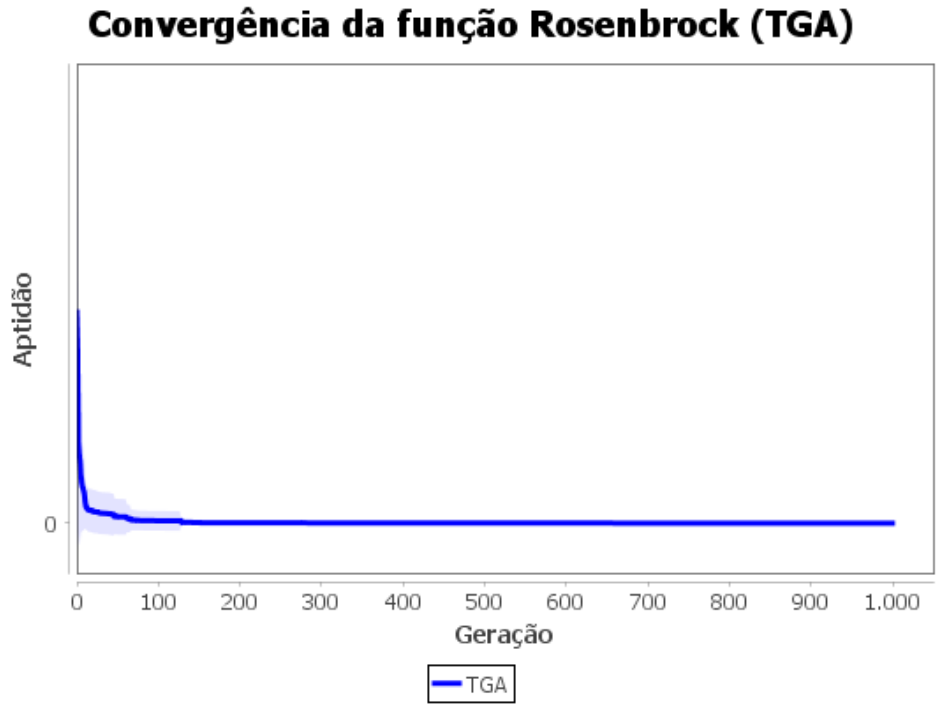
Fonte: O Autor

Figura 22 – Gráfico de Convergencia TGA da Função Rastrigin.



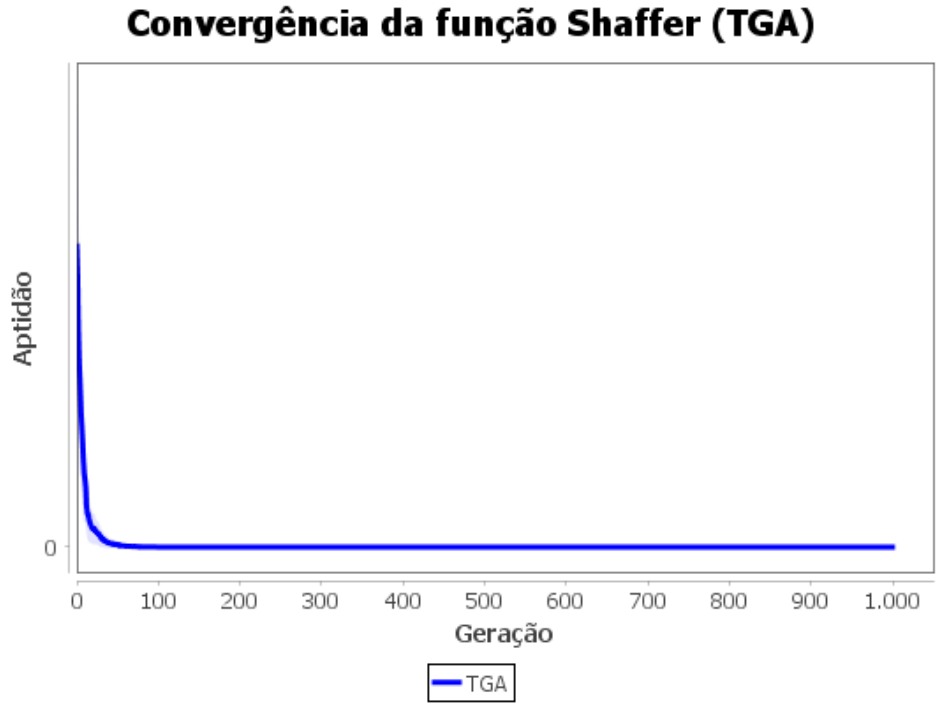
Fonte: O Autor

Figura 23 – Gráfico de Convergencia TGA da Função Rosenbrock Valley.



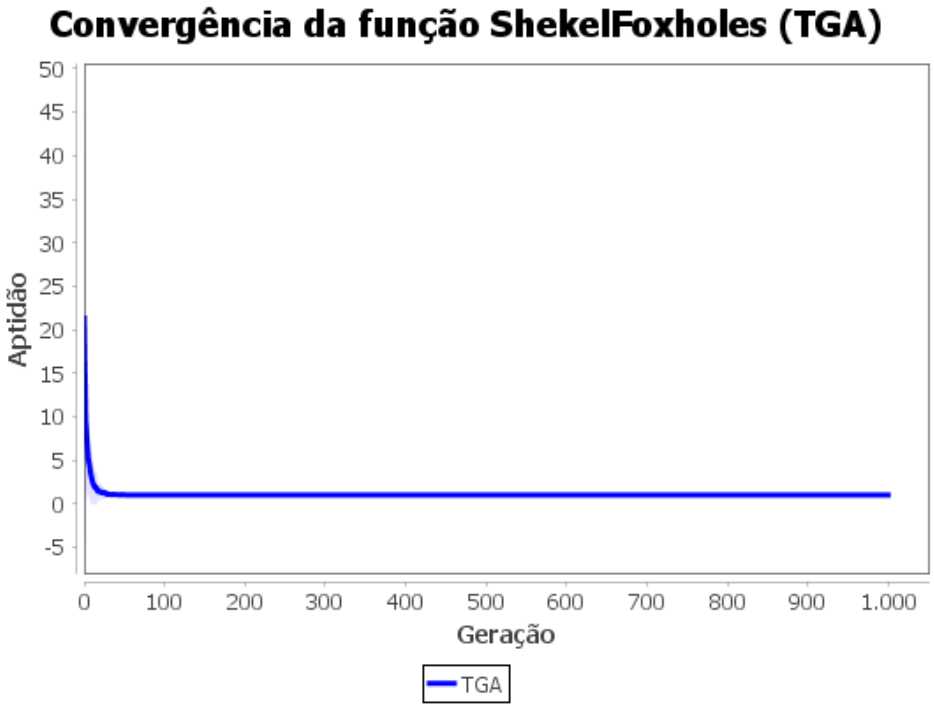
Fonte: O Autor

Figura 24 – Gráfico de Convergencia TGA da Função Shaffer.



Fonte: O Autor

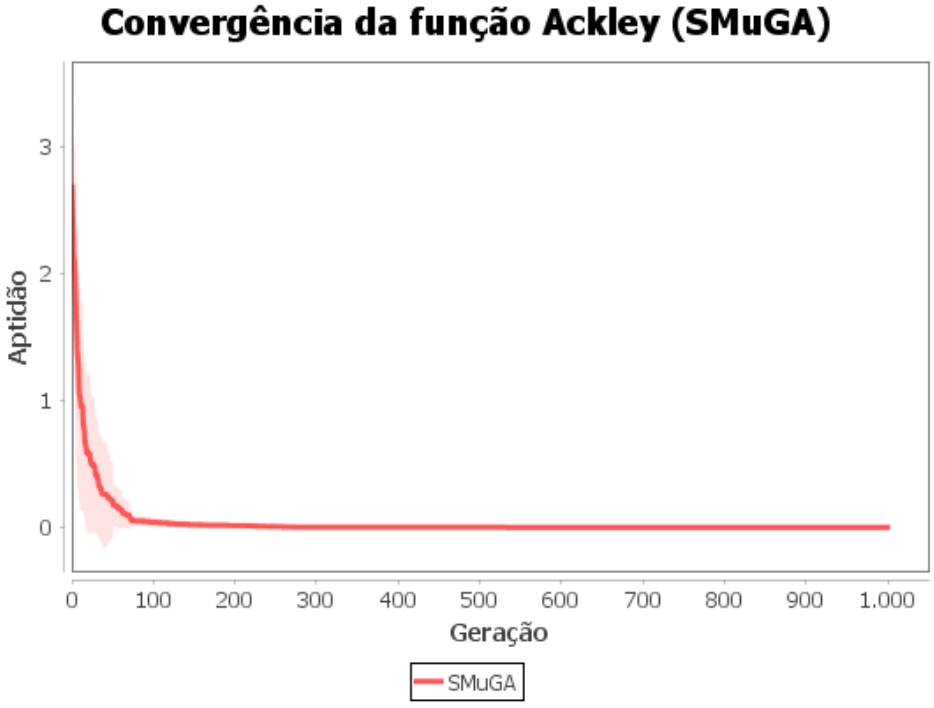
Figura 25 – Gráfico de Convergencia TGA da Função Shekel's Foxholes.



Fonte: O Autor

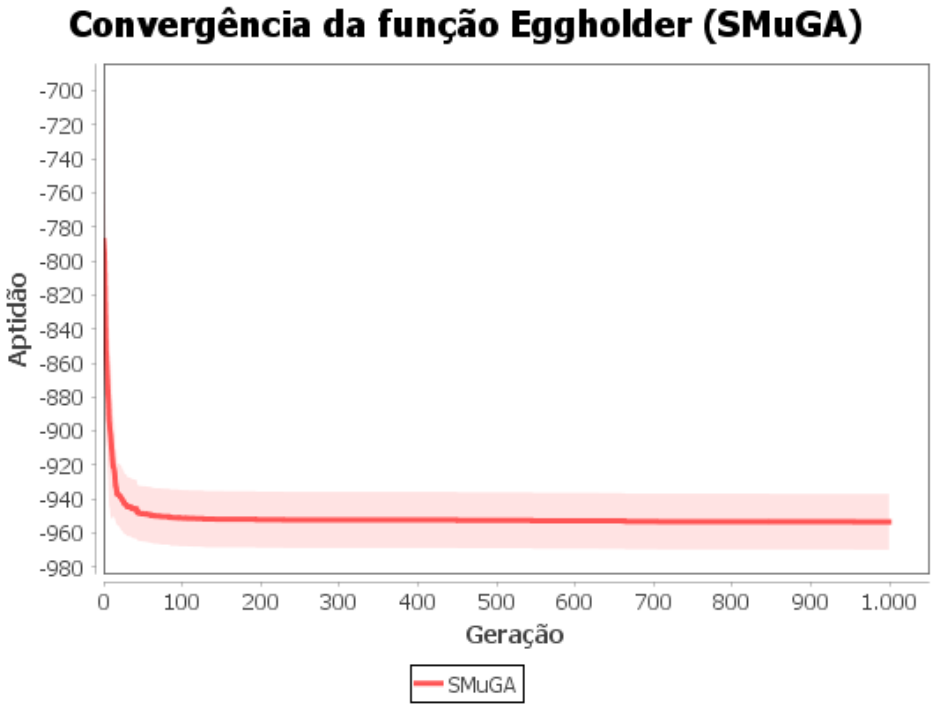
A.2 Gráficos de Convergência para o operador SMuGA

Figura 26 – Gráfico de Convergencia SMuGA da Função Ackley.



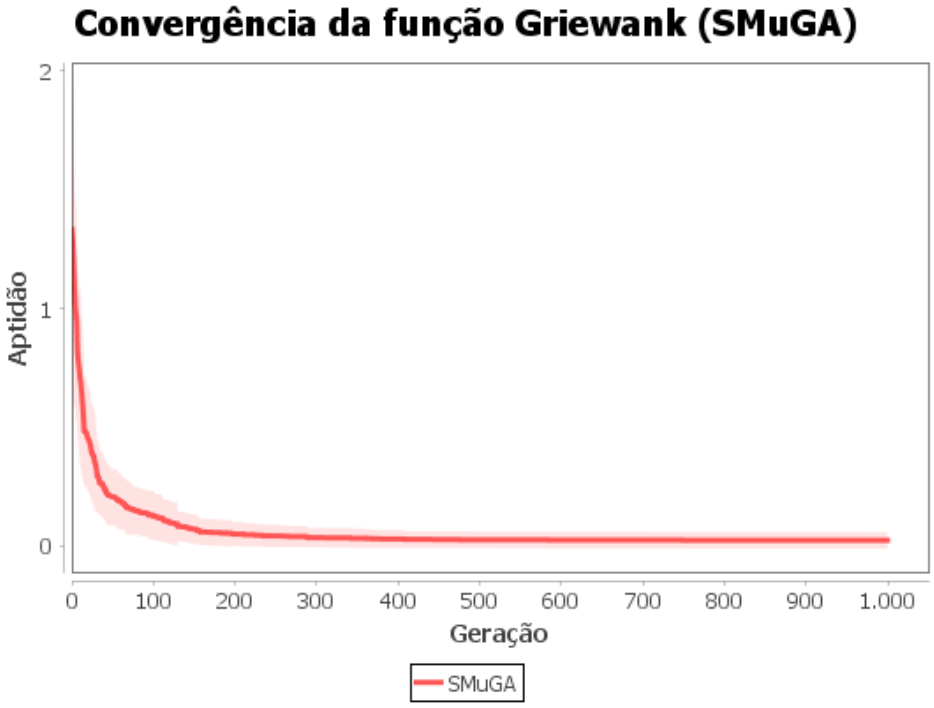
Fonte: O Autor

Figura 27 – Gráfico de Convergencia SMuGA da Função Eggholder.



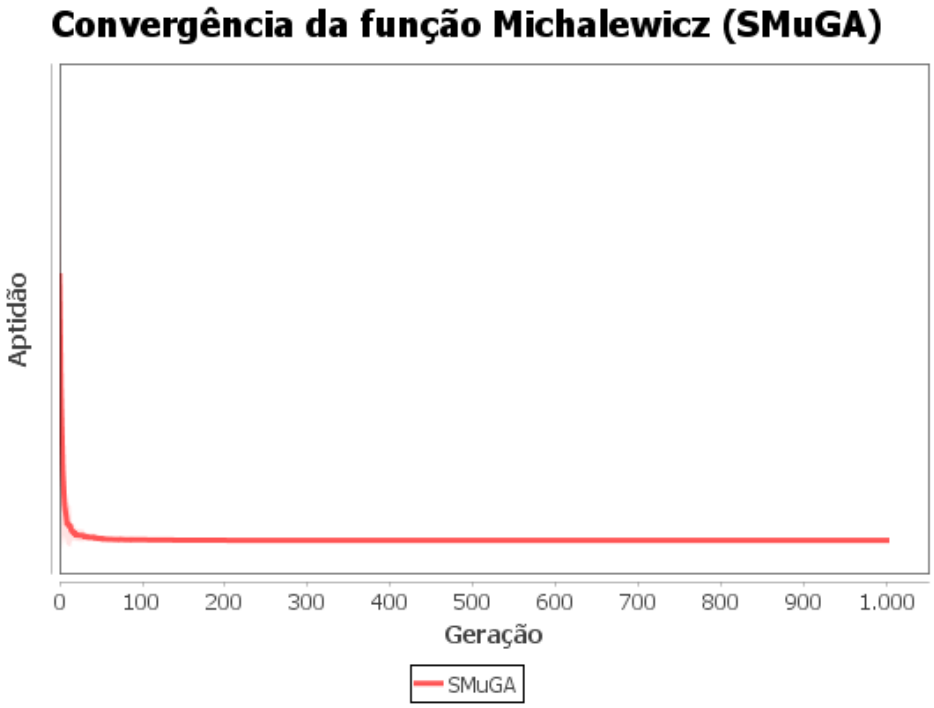
Fonte: O Autor

Figura 28 – Gráfico de Convergencia SMuGA da Função Griewank.



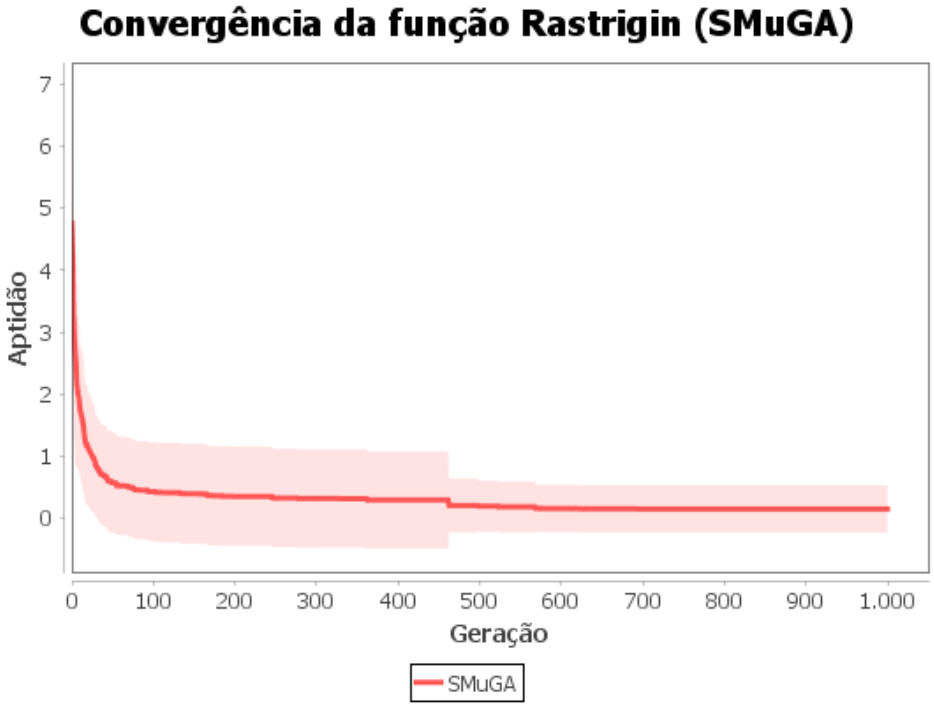
Fonte: O Autor

Figura 29 – Gráfico de Convergencia SMuGA da Função Michaelwicz.



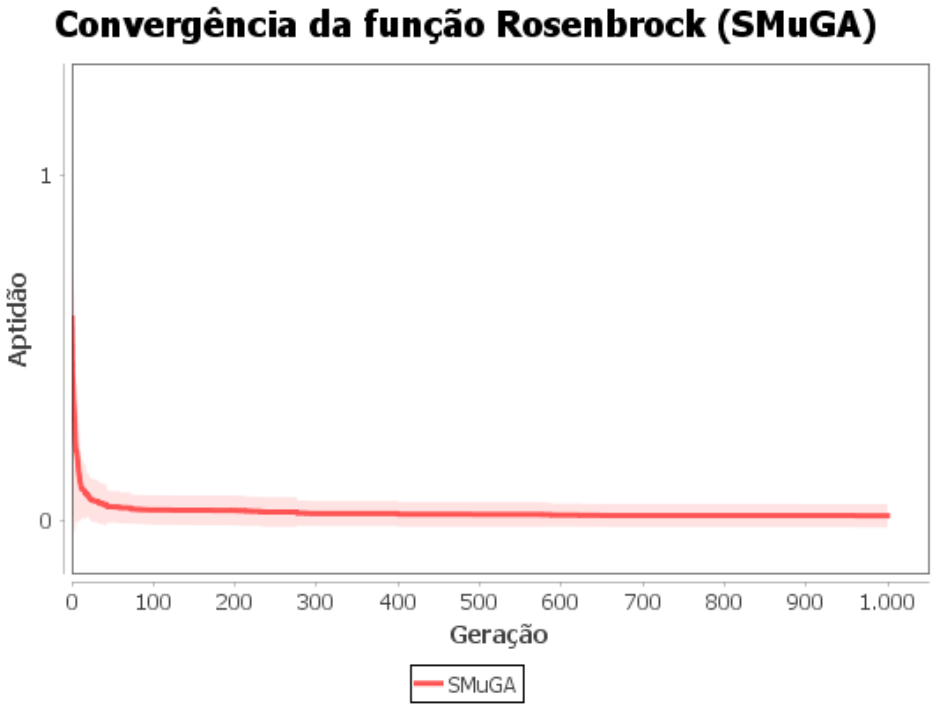
Fonte: O Autor

Figura 30 – Gráfico de Convergencia SMuGA da Função Rastrigin.



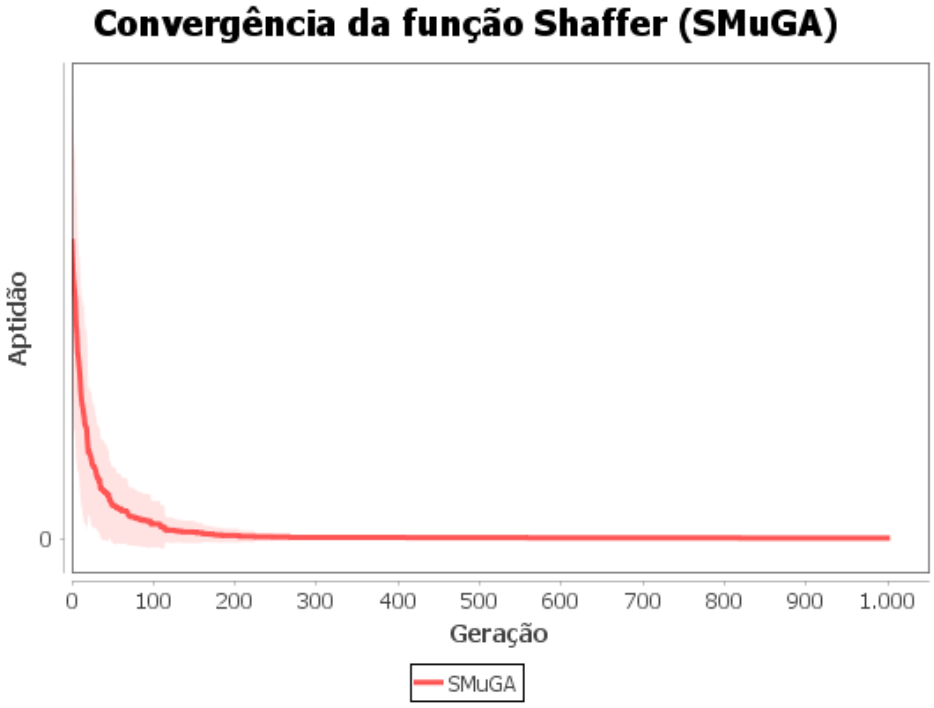
Fonte: O Autor

Figura 31 – Gráfico de Convergencia SMuGA da Função Rosenbrock Valley.



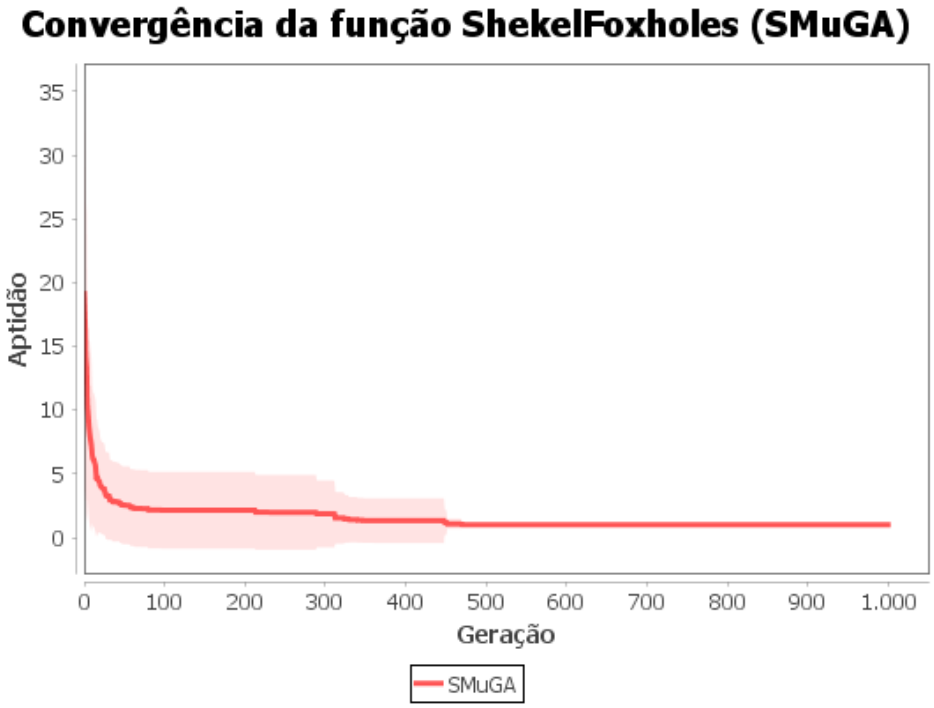
Fonte: O Autor

Figura 32 – Gráfico de Convergencia SMuGA da Função Shaffer.



Fonte: O Autor

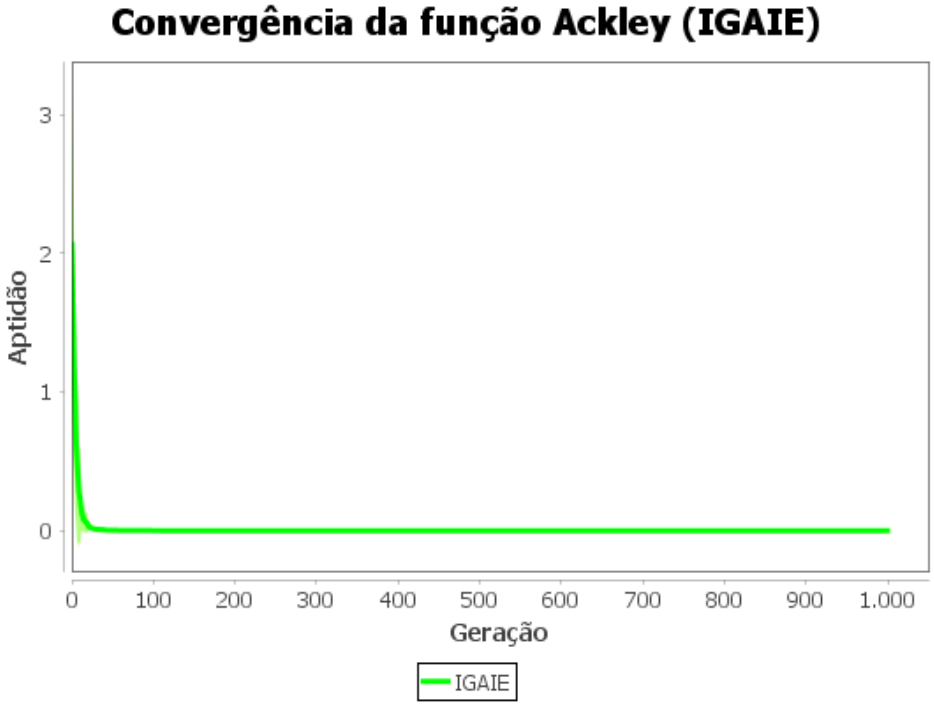
Figura 33 – Gráfico de Convergencia SMuGA da Função Shekel's Foxholes.



Fonte: O Autor

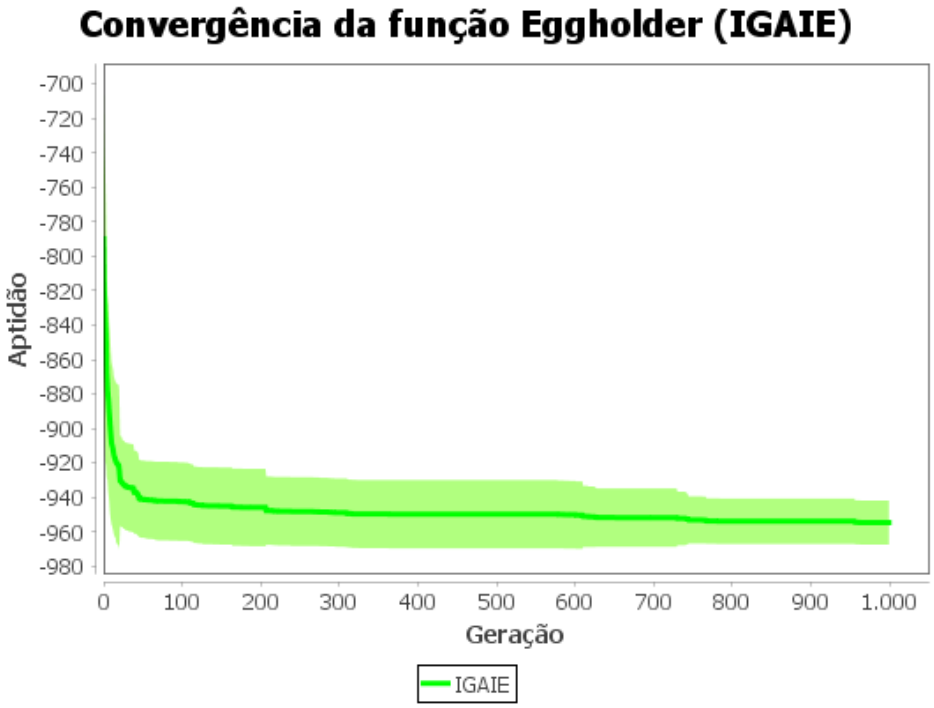
A.3 Gráficos de Convergência para o operador IGAIE

Figura 34 – Gráfico de Convergencia IGAIE da Função Ackley.



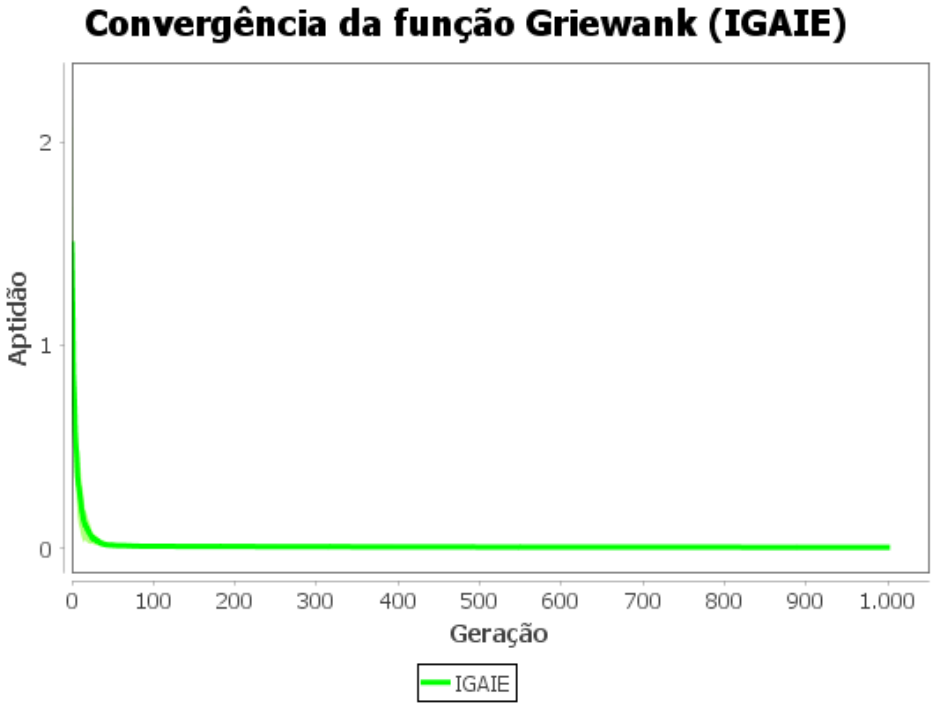
Fonte: O Autor

Figura 35 – Gráfico de Convergencia IGAIE da Função Eggholder.



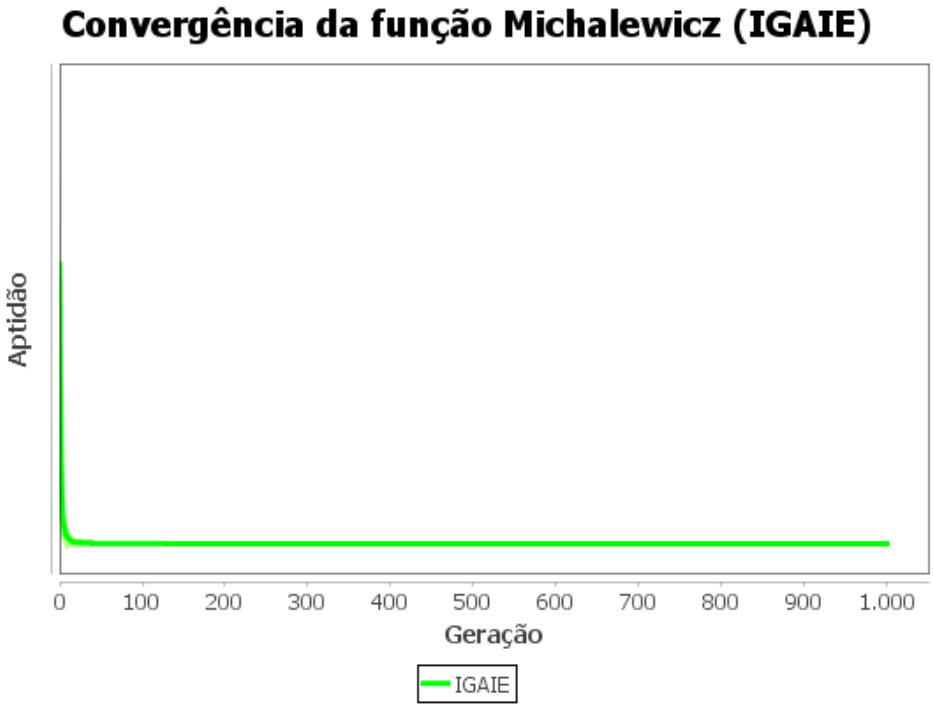
Fonte: O Autor

Figura 36 – Gráfico de Convergencia IGAIE da Função Griewank.



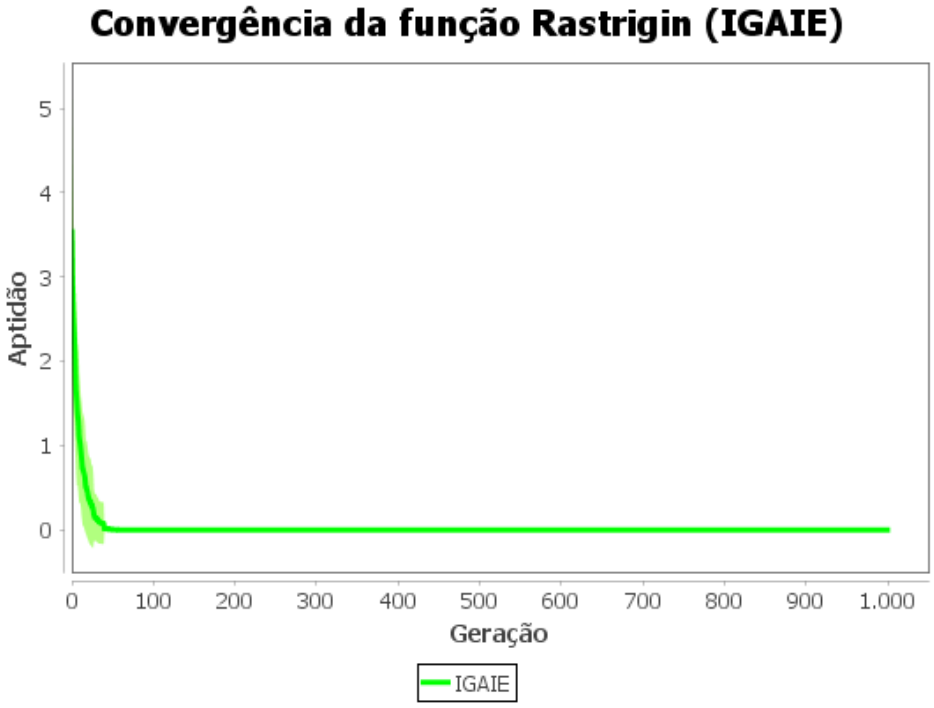
Fonte: O Autor

Figura 37 – Gráfico de Convergencia IGAIE da Função Michaelwicz.



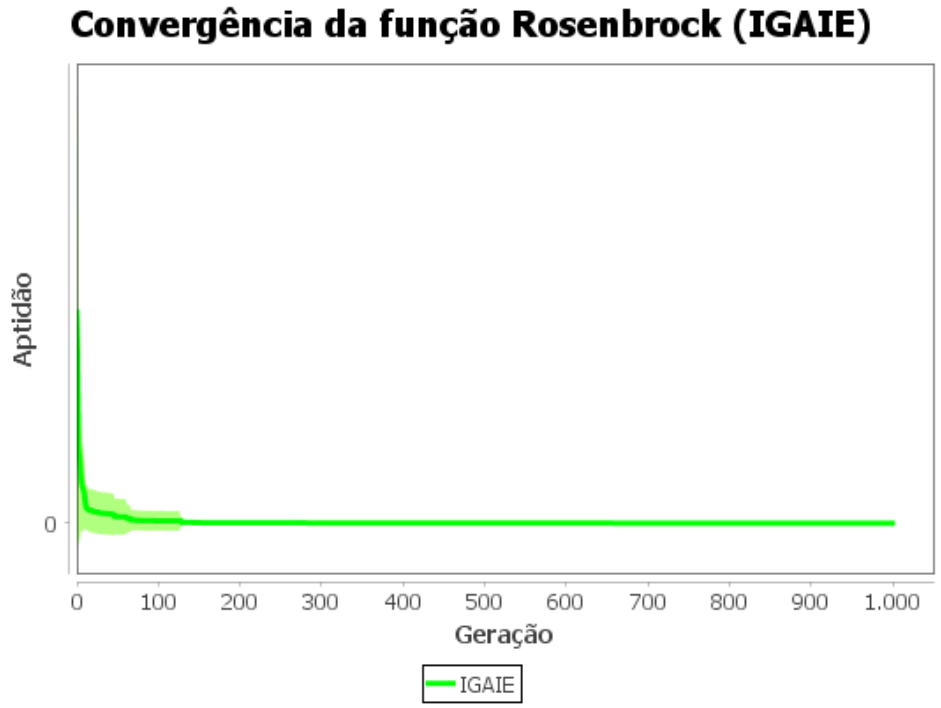
Fonte: O Autor

Figura 38 – Gráfico de Convergencia IGAIE da Função Rastrigin.



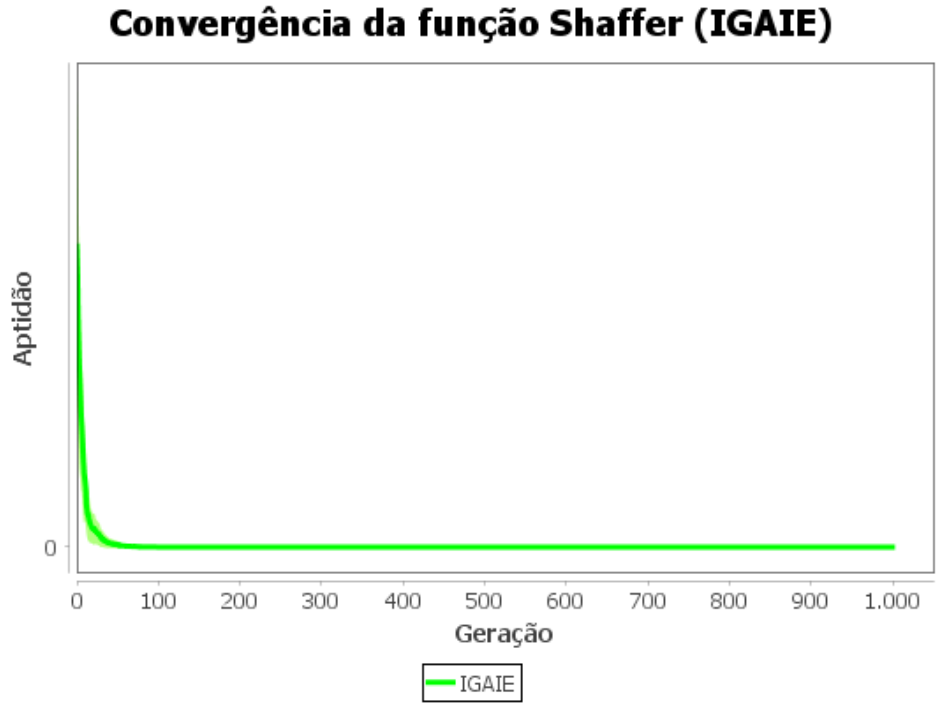
Fonte: O Autor

Figura 39 – Gráfico de Convergencia IGAIE da Função Rosenbrock Valley.



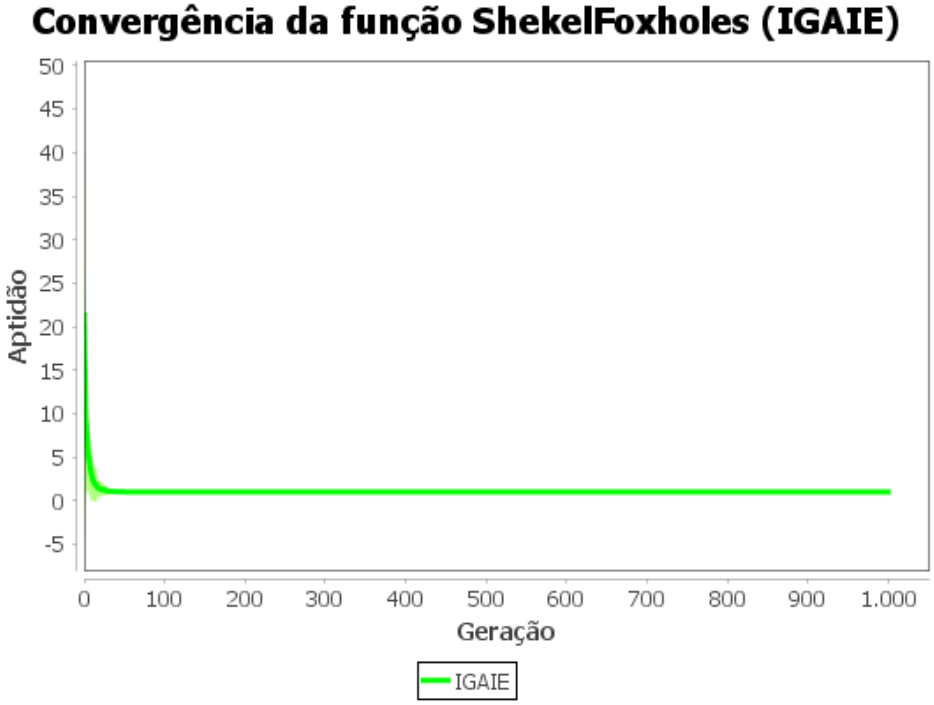
Fonte: O Autor

Figura 40 – Gráfico de Convergencia IGAIE da Função Shaffer.



Fonte: O Autor

Figura 41 – Gráfico de Convergencia IGAIE da Função Shekel's Foxholes.

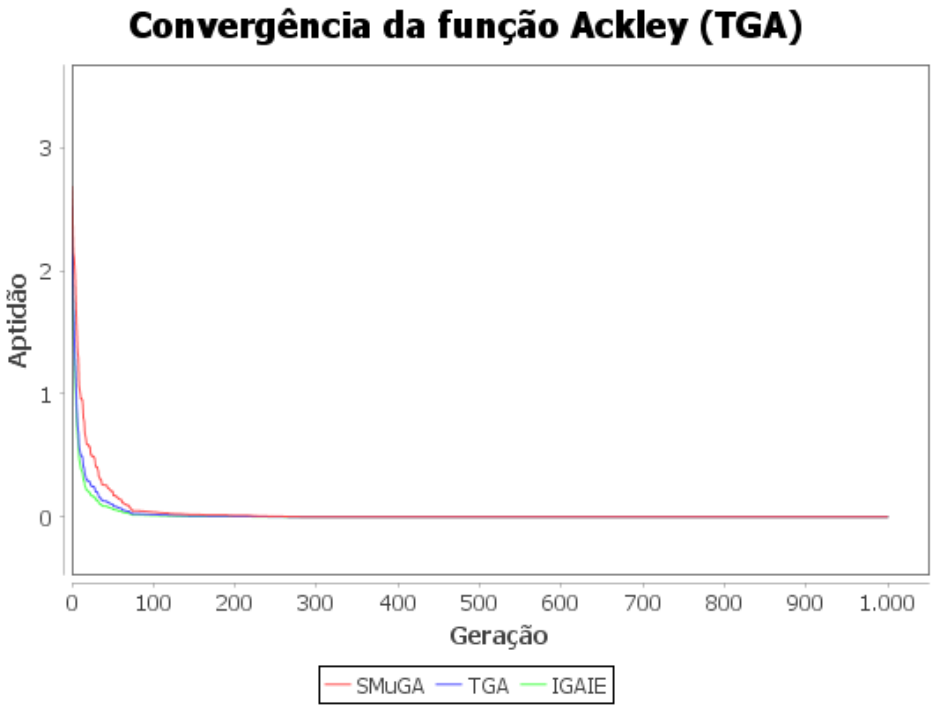


Fonte: O Autor

APÊNDICE B – GRÁFICOS DE COMPARAÇÃO DAS FUNÇÕES

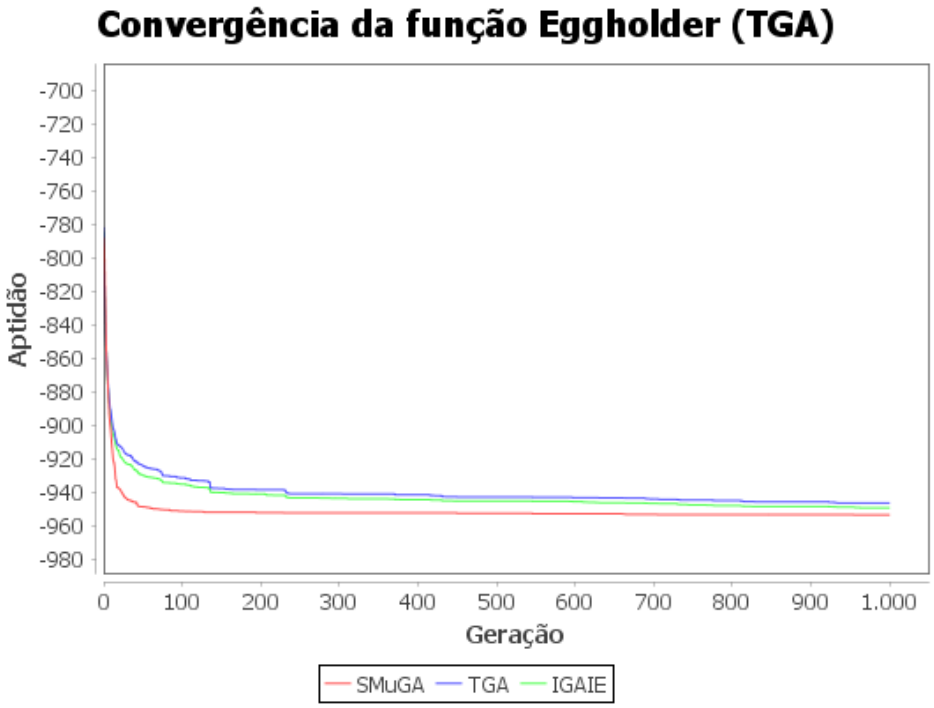
B.1 Gráficos das Funções

Figura 42 – Comparação de Convergência da Função Ackley.



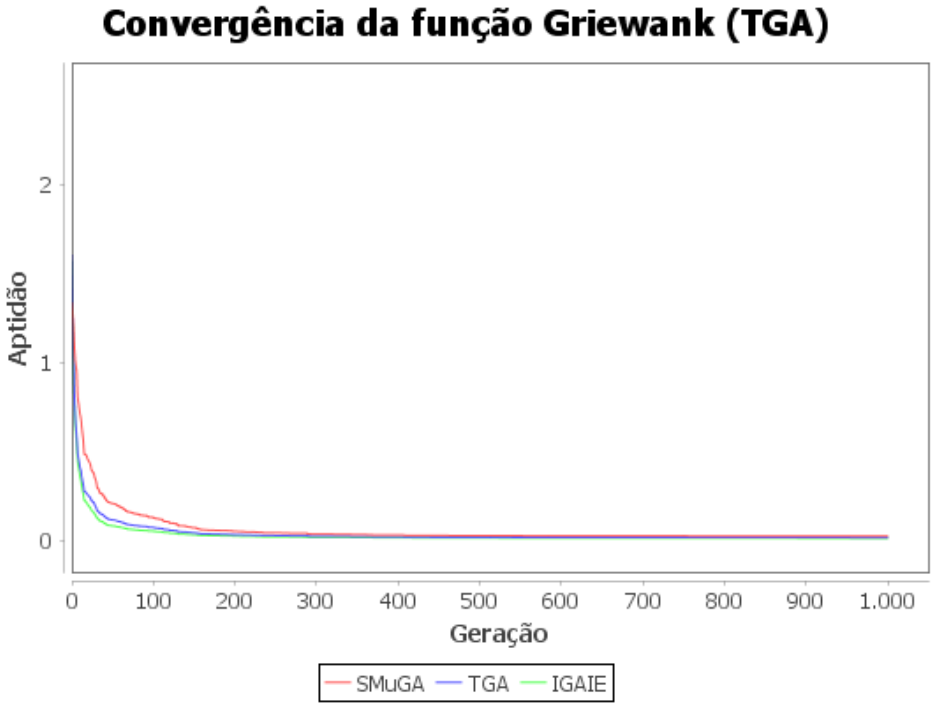
Fonte: O Autor

Figura 43 – Comparação de Convergência da Função Eggholder.



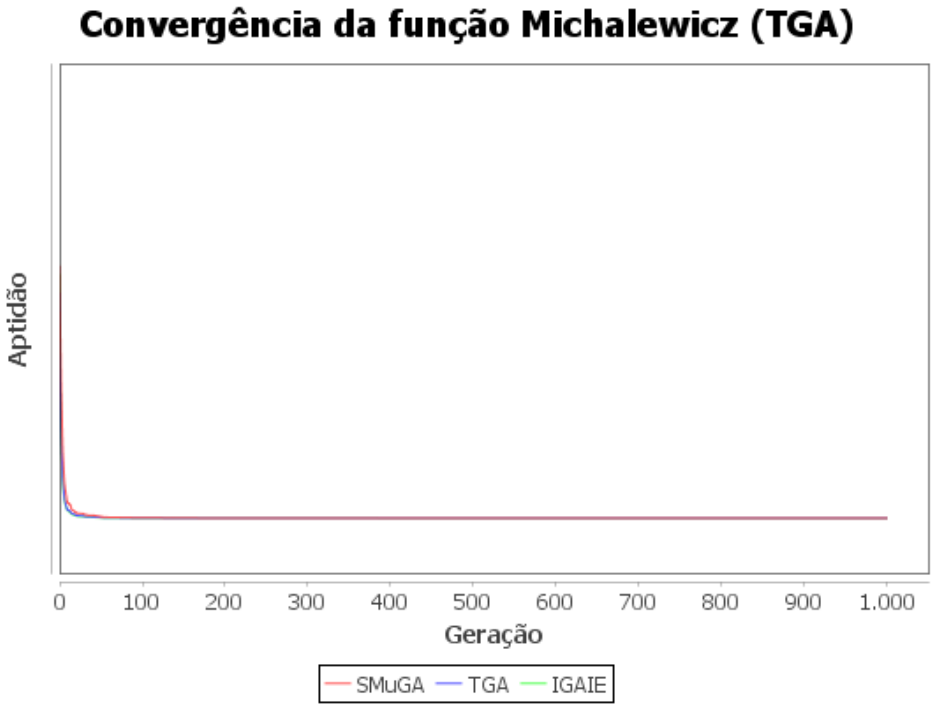
Fonte: O Autor

Figura 44 – Comparação de Convergência da Função Griewank.



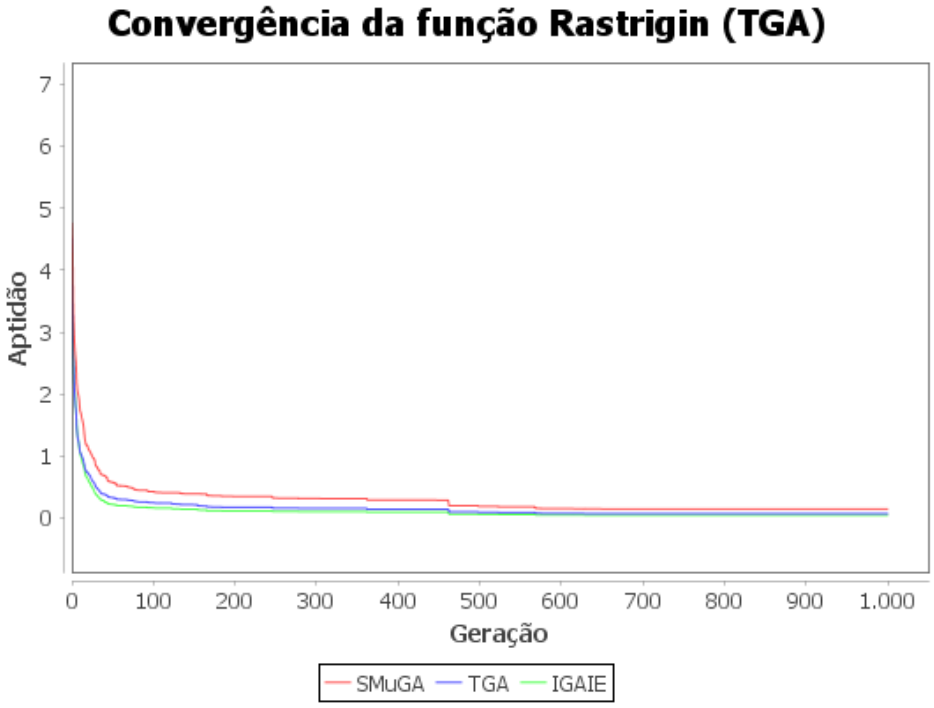
Fonte: O Autor

Figura 45 – Comparação de Convergência da Função Michalewicz.



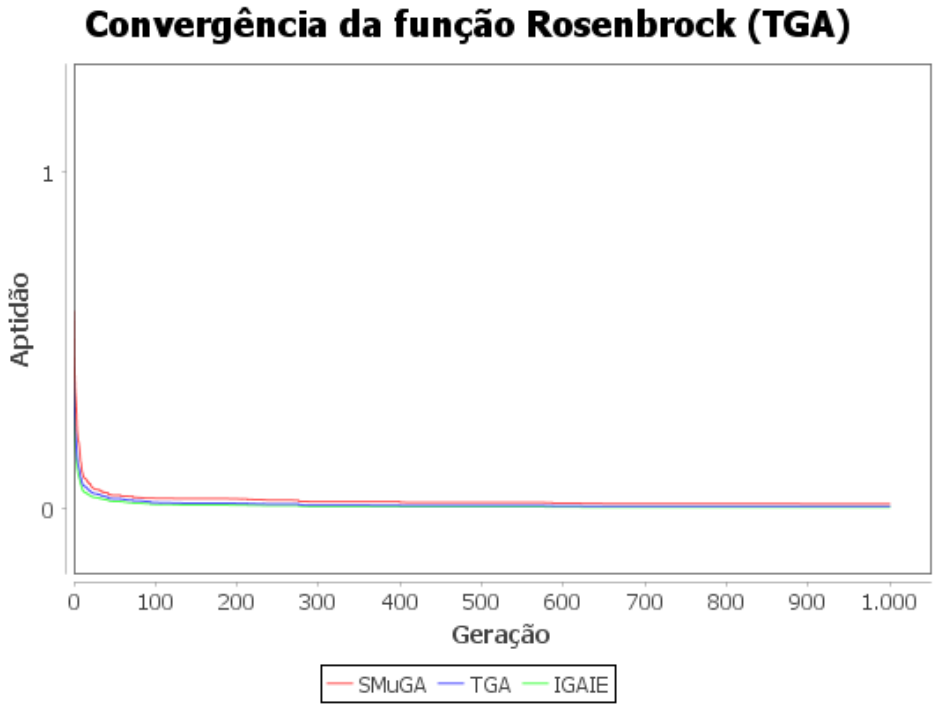
Fonte: O Autor

Figura 46 – Comparação de Convergência da Função Rastrigin.



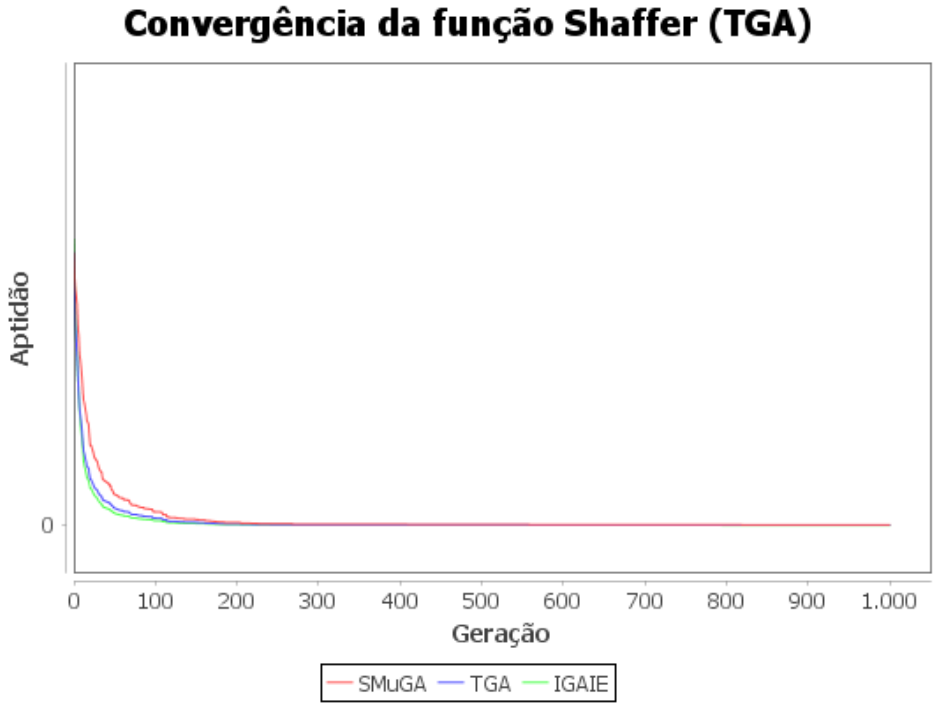
Fonte: O Autor

Figura 47 – Comparação de Convergência da Função Rosenbrock.



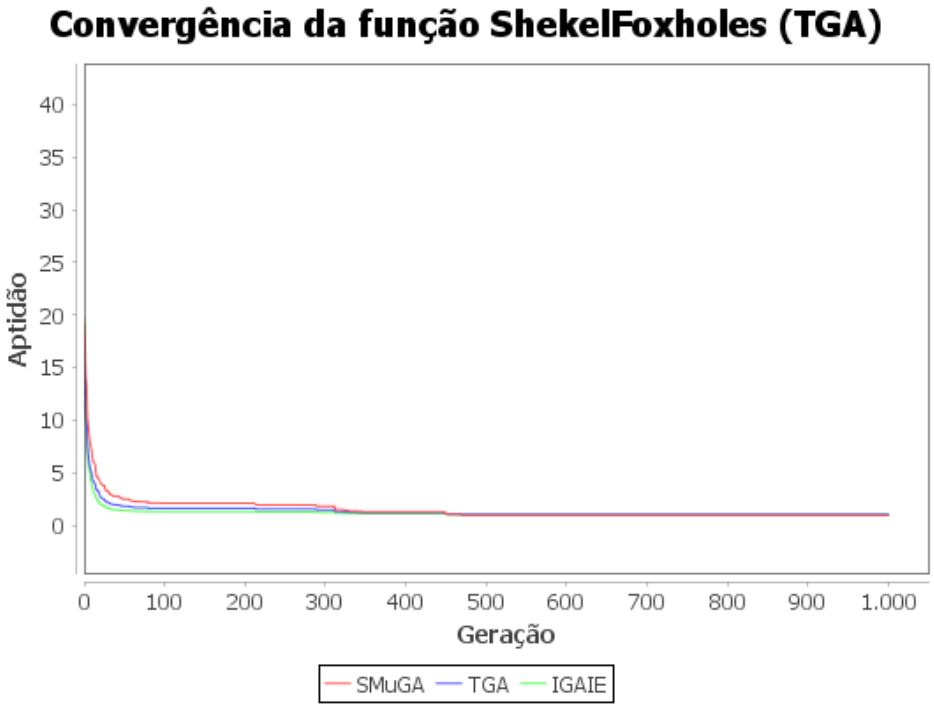
Fonte: O Autor

Figura 48 – Comparação de Convergência da Função Schaffer.



Fonte: O Autor

Figura 49 – Comparação de Convergência da Função Shekel Foxholes.

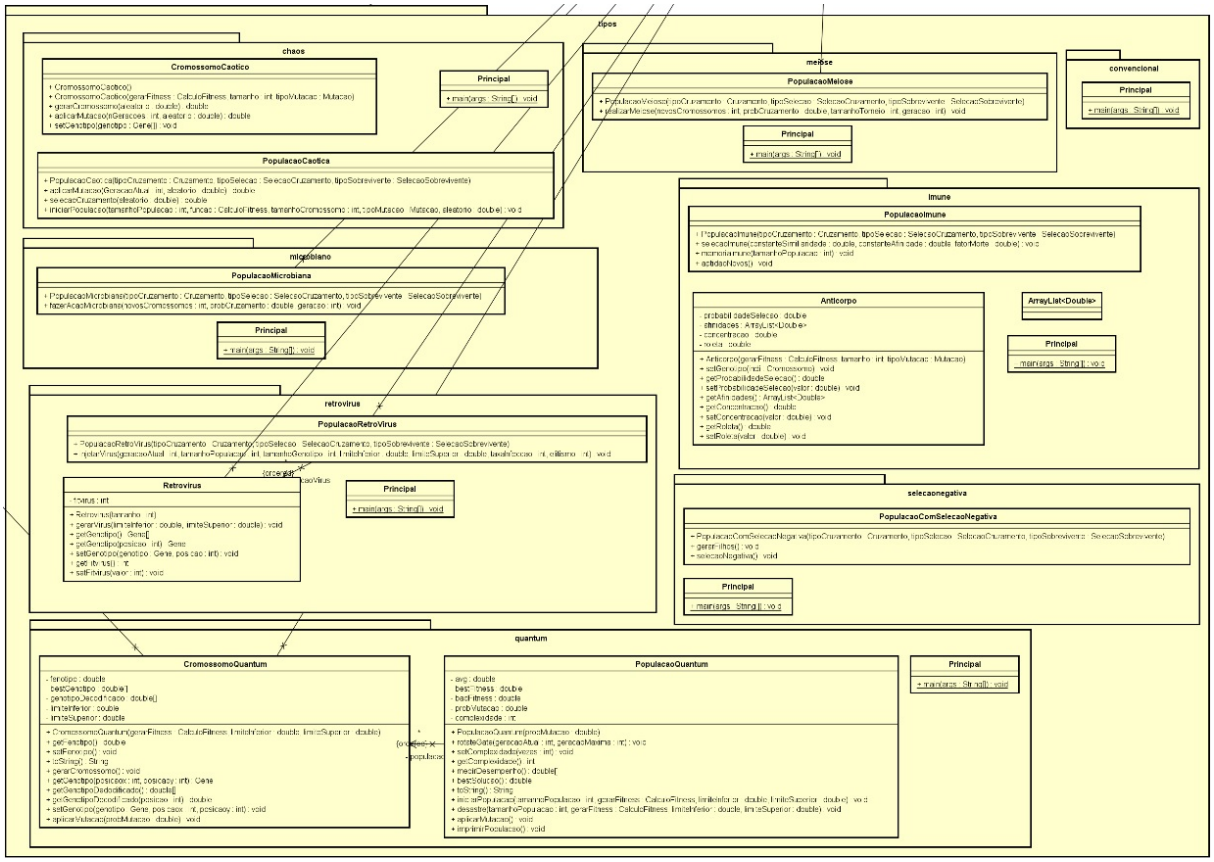


Fonte: O Autor

APÊNDICE C – UML

C.1 UML de Classes Framework

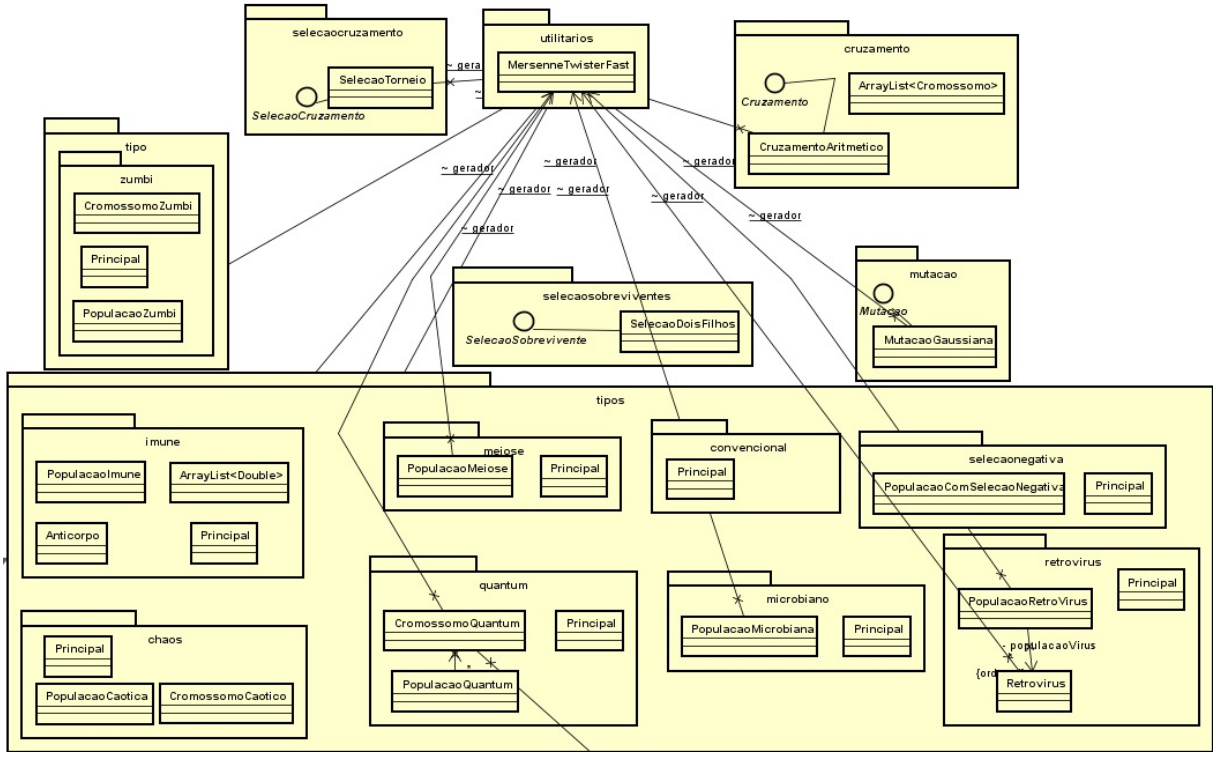
Figura 50 – UML de Classes sem Conexões(Parte dos Operadores).



Fonte: O Autor

Esta Figura ilustra somente o geral da parte dos operadores implementados.

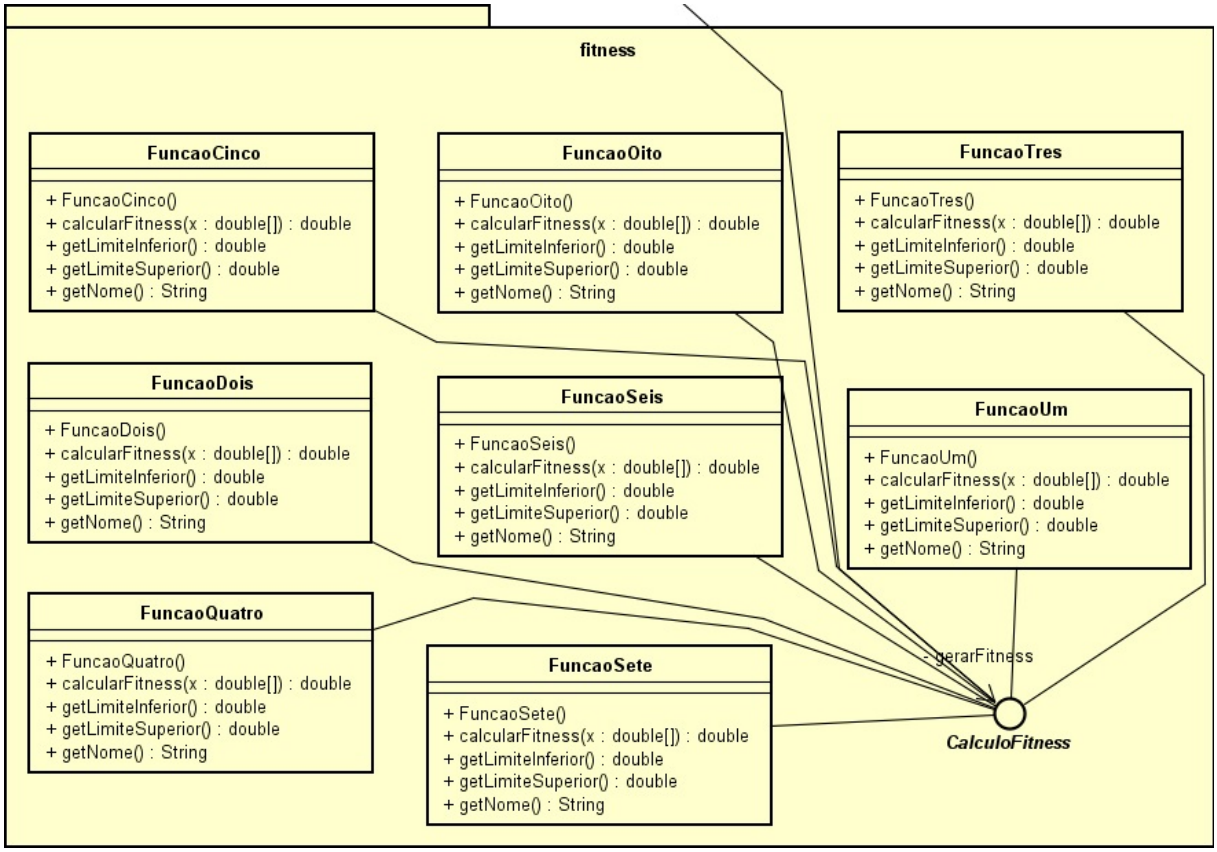
Figura 51 – UML do de Classes Completa (Parte do AG e Operadores).



Fonte: O Autor

Esta Figura mostra as relações entre classes de diferentes pacotes.

Figura 52 – UML do Framework Utilizado (Parte das Funções de Teste).



Fonte: O Autor

Esta Figura mostra as funções de teste que foram utilizadas, cada função foi executada através do pacote utilitario mostrado na Figura anterior.