Impactos do Operador Transgênico na Otimização de Funções Matemáticas Utilizando Algoritmos Genéticos

Guilherme Antonio Coelho Neto

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2025

Guilherme Antonio Coelho Neto

Impactos do Operador Transgênico na Otimização de Funções Matemáticas Utilizando Algoritmos Genéticos

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Laurence Rodrigues do Amaral

Trata-se da versão corrigida da dissertação. A versão original se encontra disponível na FACOM/UFU que aloja o Programa de Pós-Graduação em Ciência da Computação.

Uberlândia
2025

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

C672
2025

Coelho Neto, Guilherme Antônio, 1988-
Impactos do Operador Transgênico na Otimização de Funções Matemáticas Utilizando Algoritmos Genéticos [recurso eletrônico] / Guilherme Antônio Coelho Neto. - 2025.

Orientador: Laurence Rodrigues do Amaral.
Dissertação (Mestrado) - Universidade Federal de Uberlândia, Pós-graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.di.2025.164
Inclui bibliografia.

1. Computação. I. Amaral, Laurence Rodrigues do,1978-, (Orient.). II. Universidade Federal de Uberlândia. Pós-graduação em Ciência da Computação. III. Título.

CDU: 681.3

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:

Gizele Cristine Nunes do Couto - CRB6/2091
Nelson Marcos Ferreira - CRB6/3074

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Computação
Av. João Naves de Ávila, 2121, Bloco 1A, Sala 243 - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4470 - www.ppgco.facom.ufu.br - cpgfacom@ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Ciência da Computação				
Defesa de:	Dissertação, 05/2025, PPGCO				
Data:	27 de fevereiro de 2025	Hora de início:	09:02	Hora de encerramento:	11:46
Matrícula do Discente:	12212CCP009				
Nome do Discente:	Guilherme Antônio Coelho Neto				
Título do Trabalho:	Impactos do Operador Transgênico na Otimização de Funções Matemáticas Utilizando Algoritmos Genéticos				
Área de concentração:	Ciência da Computação				
Linha de pesquisa:	Inteligência Artificial				
Projeto de Pesquisa de vinculação:	----------				

Reuniu-se por videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Ciência da Computação, assim composta: Professores Doutores: Pedro Luiz Lima Bertarini - FEELT/UFU , Edimilson Batista dos Santos - CCOMP/UFSJ e Laurence Rodrigues do Amaral - FACOM/UFU , orientador do candidato.

Os examinadores participaram desde as seguintes localidades: Edimilson Batista dos Santos - São João Del-Dei/MG, Laurence Rodrigues do Amaral, Pedro Luiz Lima Bertarini e o aluno Guilherme Antônio Coelho Neto participaram da cidade Patos de Minas/MG.

Iniciando os trabalhos o presidente da mesa, Prof. Dr. Laurence Rodrigues do Amaral, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao Discente a palavra para a exposição do seu trabalho. A duração da apresentação da Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir o senhor presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir ao candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

Aprovado

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre.

O competente diploma será expedido após cumprimento dos demais requisitos,

conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Pedro Luiz Lima Bertarini**, **Professor(a) do Magistério Superior**, em 27/02/2025, às 14:38, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Edimilson Batista dos Santos**, **Usuário Externo**, em 27/02/2025, às 14:39, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Laurence Rodrigues do Amaral**, **Professor(a) do Magistério Superior**, em 27/02/2025, às 14:39, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **6100076** e o código CRC **38403F7D**.

Referência: Processo nº 23117.009395/2025-61 SEI nº 6100076

Este trabalho é dedicado a todas as pessoas que, assim como eu, acreditam na ciência e sua capacidade de transformar o mundo em um lugar melhor, mais sustentável e equitativo em oportunidades.

Agradecimentos

Agradecer é se lembrar de pessoas importantes, que apoiam e que fazem parte das realizações da vida. Eu gostaria de destacar não só as pessoas que são presentes e que fazem parte diariamente da minha vida, mas todos que contribuíram positivamente para que eu pudesse realizar este trabalho. A todos que citarei, agradeço imensamente e na mesma proporção.

Primeiramente, agradeço a **Deus**, pelo dom da vida, pela minha capacidade de discernimento e todas as dádivas e felicidades da vida que ele me presenteou.

A meus pais, **Aparecida** e **Vicente**, pela educação que me deram, pelos valores que me ensinaram, por toda a sua dedicação e cuidados durante a vida, por serem a minha base e modelo de dedicação, perseverança, respeito, esperança, amor e luta. Agradeço também as minhas irmãs, sobrinhas e todo o restante da minha família por estarem presentes nos momentos importantes e felizes da minha vida, pois me ajudaram muito a vencer desafios para que eu chegasse até aqui.

Deixo um agradecimento mais que especial à minha namorada **Valéria Costa Dias**. Pela sua paciência, compreensão e dedicação em me ver feliz e motivado diante dos desafios. Assim como os estudos me fazem me sentir um homem melhor e mais realizado, o seu amor me faz uma pessoa melhor a cada dia. Obrigado por caminhar ao meu lado.

Ao **Prof. Dr. Laurence Rodrigues do Amaral** pela sua orientação, dedicação, paciência e profissionalismo. Todo o apoio e atenção dedicados a mim foram imprescindíveis para a realização deste trabalho. Agradeço também aos membros da banca avaliadora, aos quais suas recomendações e observações me ajudarão a evoluir e progredir no meu caminho da pesquisa científica. Agradeço também a todos os professores do Programa de Pós-graduação em Ciência da Computação (PPGCO) - UFU que me proporcionaram inúmeros aprendizados que contribuíram com minha pesquisa e construção como pesquisador e ser humano.

Por fim, agradeço a todos os meus amigos, colegas de trabalho e colegas do PPGCO-UFU que estiveram comigo nessa minha caminhada.

"O cientista não é o homem que fornece as verdadeiras respostas;
e sim, quem faz as verdadeiras perguntas."
(Claude Lévi-Strauss)

Resumo

Algoritmos Genéticos são ferramentas metaheurísticas de busca e otimização inspirados nos princípios da genética e da evolução. Variações do Algoritmo Genético surgem com o intuito de agilizar a descoberta de uma solução ótima e reduzir custos computacionais. O Algoritmo Genético Transgênico é uma variação do Algoritmo Genético convencional que incorpora os conceitos da transgênese, através do operador genético *transgenic*. Com o intuito de investigar o seu potencial em melhorar a acurácia e a qualidade de convergência na otimização de fórmulas matemáticas com algoritmos genéticos, esta pesquisa avalia as respectivas vantagens e desvantagens do uso do Algoritmo Genético Transgênico e do Algoritmo Genético Convencional na otimização de fórmulas matemáticas com múltiplos ótimos locais. Os resultados dos experimentos demonstraram que o operador transgênico aumentou a capacidade de convergência e reduziu o tempo de processamento na otimização de fórmulas matemáticas utilizando algoritmos genéticos.

Palavras-chave: Algoritmos Genéticos. Operador Transgênico. Computação Evolutiva. Otimização de Funções Matemáticas.

Abstract

Genetic Algorithms are metaheuristic search and optimization tools inspired by the principles of genetics and evolution. Variations of the Genetic Algorithm have emerged to quicken the discovery of optimal solutions and reduce computational costs. The Transgenic Genetic Algorithm is a variation of the Conventional Genetic Algorithm that incorporates the concepts of transgenesis through the *transgenic* genetic operator. In order to investigate its potential to improve convergence accuracy and quality in the optimization of mathematical formulas with genetic algorithms, this research evaluates the respective advantages and disadvantages of using the Transgenic Genetic Algorithm and the Conventional Genetic Algorithm in the optimization of mathematical formulas with multiple local optima. The results of the experiments demonstrated that the *transgenic* operator increased the convergence capacity and reduced the processing time in the optimization of mathematical formulas using genetic algorithms.

Keywords: Genetic Algorithms. Transgenic Operator. Evolutionary Computation. Mathematical Function Optimization.

Lista de ilustrações

Lista de tabelas

Lista de algoritmos

Lista de códigos

Lista de siglas

AG Algoritmo Genético

API Application Programming Interface Library

BIN-NLCEE Binary Nonlinear Computational Evolutionary Environment

CA Cellular Automata

CE Computação Evolutiva

CEC Congress on Evolutionary Computation

CEE Computational Evolutionary Environment

COPS Combinatorial Optimization Problems

EE Estratégias Evolutivas

GO Gene Ontology

IA Inteligência Artificial

IG Information Gain

IG-CEE Information Gain-based Computational Evolutionary Environment

JSSP Job-Shop Schedulig Problem

MDRGA Multiple Disjunctions Rule Genetic Algorithm

MRS Multi-Robot Systems

NLCEE Non-linear Computation Evolutionary Environment

OGM Organismo Geneticamente Modificado

PG Programação Genética

PE Programação Evolutiva

SC Sistemas de Classificação

Lista de símbolos

$\in$	Set membership (belongs to) symbol
π	Pi mathematical constant
$\sum$	Summation operator symbol
f	Function denotation

Mathematical constants

π	Pi constant

Mathematical symbols

$\in$	Set membership (belongs to)
$\sum$	Summation operator
f	Function denotation

Sumário

Capítulo 1

Introdução

A necessidade de encontrar soluções para diferentes níveis e tipos de adversidades fez com que a humanidade buscasse soluções cada vez mais criativas e adaptadas às particularidades dos problemas. Na natureza, as ciências humanas encontraram uma inesgotável fonte de inspiração para abordagens solucionistas. Inspirada nos princípios da Evolução e da Biogenética, a Computação Evolutiva (CE) despontou como uma subárea da Inteligência Artificial (IA) composta por um conjunto de técnicas e paradigmas destinados à resolução de problemas computacionais (ERTEL, 2017). Entre as principais técnicas que compõem a CE, estão os Algoritmos Genéticos (Algoritmo Genético (AG)s). Os AGs são mais indicados para problemas NP-Completos, com múltiplas variáveis e espaços de busca grandes e complexos, onde algoritmos convencionais e determinísticos não são indicados (MAN, 1996). Contudo, os Algoritmos Genéticos também possuem suas limitações. Os AGs possuem baixa taxa de convergência em certos tipos de problemas (como por exemplo, funções matemáticas complexas). Estes tipos de problemas podem apresentar vastos espaços de busca, múltiplos picos e ótimos locais de difícil localização, que diminuem as probabilidades de convergência (ARORA, 2024).

Baseados em aspectos bioevolutivos, os Algoritmos Genéticos são altamente adaptativos, possuem a capacidade de absorver novos princípios para aprimorar sua eficácia (MITCHELL, 1996). Novos modelos ou variações de Algoritmos Genéticos são implementados para se adaptarem a diferentes classes de problemas com complexidades variáveis. O Algoritmo Genético Transgênico é uma variação de Algoritmo Genético que inclui os benefícios da transgenia. A transgenia permite a produção dos Organismos Geneticamente Modificados (OGMs): espécies de indivíduos (geralmente plantas ou animais) obtidos através de processos de melhoramento genético (XAVIER, 2008). O processo de transgenia envolve a recepção de genes pré-selecionados de outros organismos para expressar características desejáveis (VARGAS, 2018). Desse modo, a transgenia oferece vantagens consideráveis, pois realiza uma seleção elitista de genes que carregam características desejáveis a serem mantidas nas próximas gerações.

Se inspirando nos conceitos e benefícios encontrados na transgenia, Amaral e Hrus-

chka (2011) propuseram um novo operador genético para Algoritmos Genéticos, o operador *transgenic*. De forma similar ao processo de transgenia do mundo real, o operador *transgenic* identifica genes específicos que são relevantes em indivíduos com alto valor de aptidão e introduz as características e informações presentes nesses genes nos genes de indivíduos com baixo valor de aptidão (AMARAL; HRUSCHKA, 2011). O intuito é acelerar o processo de otimização e reduzir os custos computacionais. Diante do potencial promissor do operador *transgenic*, este trabalho propõe um estudo investigativo sobre a acurácia e performance do Algoritmo Genético Transgênico e do Algoritmo Genético convencional na otimização de fórmulas matemáticas de diferentes características e complexidades.

1.1 Motivação

Os Algoritmos Genéticos são métodos heurísticos para tratar problemas que não necessitam obrigatoriamente do reconhecimento de uma solução ótima, mas sim uma aproximação. Eles são procedimentos versáteis para lidar com problemas de grande escala, que demandam a exploração aleatória em espaços de soluções. Porém, possuem a tendência de serem computacionalmente pesados, consumindo recursos de memória e processamento. Além do custo computacional, os AGs frequentemente apresentam uma taxa de convergência limitada ao lidar com determinados problemas, especialmente problemas que possuam espaços de busca amplos, numerosos máximos locais e ótimos locais difíceis de identificar, como funções matemáticas de maior complexidade.

Os Algoritmos Transgênicos são muito promissores. Possuem os benefícios da transgenia, que podem acelerar a busca por soluções ótimas, agilizar a fuga de ótimos locais, explorar soluções em regiões promissoras do espaço de busca e reduzir o número populacional para convergência do AG. Analisando essas possibilidades, este estudo tem a motivação de descobrir qual abordagem, entre o AG convencional ou o AG transgênico, possui melhor precisão na otimização de problemas complexos com baixa convergência.

1.2 Objetivos e Desafios da Pesquisa

Os Algoritmos Evolucionários surgiram com o desafio de otimizar problemas de diferentes graus de complexidade. Problemas de múltiplos picos locais, espaço de busca complexos, ótimos globais de difícil identificação tendem a ser problemas de elevado grau de complexidade até mesmo para os Algoritmos Genéticos. O objetivo básico é verificar a capacidade do operador transgênico em agregar valor à otimização, por meio de melhorias na acurácia, nas taxas de convergência e/ou na redução dos custos computacionais do processo de otimização de fórmulas matemáticas. Para isso, eles serão submetidos à busca de ótimos locais em diferentes funções matemáticas, que representam problemas de

diferentes níveis de complexidade e por possuírem várias das características acima citadas desse tipo de problema.

Os objetivos específicos são:

- ❏ Implementar um Algoritmo Genético convencional para a busca e otimização de mínimos e máximos globais em funções matemáticas de complexidades variáveis.
- ❏ Implementar um Algoritmo Genético Transgênico para a busca e otimização de mínimos e máximos globais em funções matemáticas de complexidades variáveis.
- ❏ Investigar os impactos de performance, capacidade de convergência e uso de recursos computacionais do Algoritmo Genético Transgênico em relação ao Algoritmo Genético convencional.
- ❏ Mensurar os benefícios e impasses na utilização do Algoritmo Genético Transgênico em comparação ao Algoritmo Genético convencional na otimização de problemas de localização e estagnação de múltiplos locais de diferentes níveis de dificuldade.

1.3 Hipótese

O operador genético *transgenic* foi proposto por Amaral e Hruschka (2011). O uso do operador *transgenic* em Algoritmos Genéticos possibilita uma nova proposta de algoritmo evolucionário, inspirado nos avanços provenientes da tecnologia do DNA recombinante, denominado Algoritmo Genético Transgênico. Espera-se que essa proposta de algoritmo evolucionário tenha desempenho equiparado ou superior em alguns critérios ao compará-lo com um Algoritmo Genético convencional na busca de soluções ótimas em diferentes contextos de situações-problema. Seguem as hipóteses a serem verificadas com esta pesquisa científica:

1. O operador de transgenia aumenta a capacidade de convergência de um Algoritmo Genético?
2. Em relação ao número de gerações, o Algoritmo Genético Transgênico é mais eficiente, ou seja, converge com menos gerações, do que um Algoritmo Genético convencional?
3. Em relação ao tempo de processamento, o Algoritmo Genético Transgênico converge em menos tempo que um Algoritmo Genético convencional?
4. O tempo de processamento do Algoritmo Genético Transgênico é menor do que o tempo de processamento do algoritmo genético convencional quando ambos não convergem?

1.4 Contribuições

O desenvolvimento deste trabalho propõe as seguintes contribuições para a comunidade científica:

- ❏ Determinar as vantagens e desvantagens do uso do operador *transgenic* na otimização de equações matemáticas com múltiplos ótimos locais.
- ❏ Determinar os impactos positivos e negativos durante a otimização utilizando o Algoritmo Genético Transgênico em comparação a otimização por Algoritmos Genéticos convencionais.
- ❏ Aprimorar as alternativas existentes de técnicas de otimização através da implementação de novos Algoritmos Genéticos direcionados especificamente a otimização de fórmulas matemáticas com múltiplos picos (ótimos locais).
- ❏ Desenvolvimento do Algoritmo Genético Transgênico, um Algoritmo Genético que inclui o uso do operador genético transgênico proposto por Amaral e Hruschka (2011).
- ❏ Implementação do Algoritmo Genético Transgênico; uma variação do Algoritmo Genético convencional que inclui o operador genético transgênico, que é baseado no operador *transgenic* de Amaral e Hruschka (2011).

1.5 Organização da Dissertação ou Tese

No Capítulo 2 são apresentados os fundamentos teóricos relacionados à pesquisa. O Capítulo 3 apresenta trabalhos correlatos ao contexto deste estudo e que influenciaram a sua produção, apontando as ideias relevantes de cada trabalho e a forma como impactaram positivamente a realização desta pesquisa. O Capítulo 4 detalha a proposta da pesquisa. No Capítulo 5 são apresentados os experimentos realizados, destacando os pontos relevantes, descrevendo com detalhes as informações úteis para a interpretação dos experimentos. Por fim, o Capítulo 6 apresenta as conclusões encontradas através dos resultados dos experimentos.

Capítulo 2

Fundamentação Teórica

Este capítulo delineia os trabalhos, técnicas e conceitos essenciais para a fundamentação da pesquisa realizada neste trabalho. A estrutura organizacional deste capítulo é dividida em três subseções.

A subseção 2.1 expõe uma visão geral da CE; descrevendo sua origem, importância, desafios e técnicas mais utilizadas. A subseção 2.2 esmiúça os conceitos fundamentais relacionados aos Algoritmos Genéticos. Os Algoritmos Genéticos constituem o alicerce para o desenvolvimento desta pesquisa; por isso, torna-se essencial compreender profundamente os conceitos, características e detalhes que formam a base teórica e prática associada a essa classe de algoritmos. Por fim, a subseção 2.3 aborda conceitos relacionados aos organismos geneticamente modificados (OGMs), à tecnologia de DNA recombinante e, principalmente, elementos relacionados à transgenia: conceitos essenciais para a compreensão do operador *transgenic*, operador genético essencial ao Algoritmo Genético Transgênico.

2.1 Computação Evolutiva

2.1.1 Modelo Evolutivo e a Seleção Natural

Segundo Darwin (1859), o processo de mudanças ocorridas ao longo do tempo em indivíduos de determinada espécie é chamado de evolução. As espécies sobreviventes são as mais evoluídas por serem as mais aptas e terem conseguido se adaptar melhor às pressões, desafios e perigos impostos pelo meio ambiente. Por outro lado, as espécies extintas foram incapazes de se adaptar e satisfazer as pressões e desafios impostos para sobreviver. Este modelo natural de sobrevivência e evolução é conhecido como **seleção natural**.

As condições ambientais determinam as características que os indivíduos precisam manifestar ou aprimorar para a sobrevivência e reprodução. Indivíduos que não possuem características favoráveis em relação ao meio em que vivem tendem a não se reproduzirem, podendo ocasionar a extinção da espécie. A teoria da seleção natural, proposta por

Darwin, parte da premissa de que as características dos organismos variam de uma forma não determinística dos pais aos seus descendentes. Esse processo é chamado individualização (PILA, 2006).

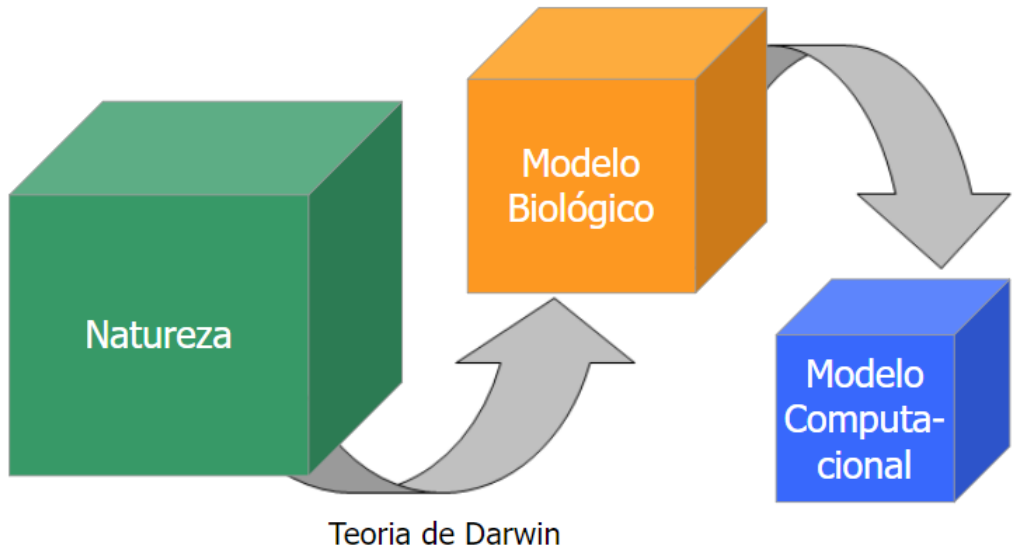
A seleção natural é o mecanismo essencial para a evolução, sobrevivência e extinção das espécies e, até o presente momento, é considerado pela comunidade acadêmica como a melhor explicação para a extinção e surgimento de novas espécies de seres vivos.

2.1.2 Origem da Computação Evolutiva

A teoria da evolução de Darwin ilustra um modelo robusto de aprimoramento e evolução das espécies através da seleção natural. A seleção natural demanda a adaptação das populações às circunstâncias específicas dos ambientes em que vivem.

Esses princípios, como a reprodução seletiva dos indivíduos mais adaptados e a perpetuação destas características em novos indivíduos que reúnam as melhores características de uma população, podem ser extrapolados para além do âmbito da biologia e serem transportados para outras situações. Situações onde não há soluções determinísticas ou situações em que se faz necessária a aplicação de soluções heurísticas para obtenção do resultado mais satisfatório possível dentre uma gama de possíveis soluções são exemplos dessa situação. Diante desse cenário, a CE surgiu como uma abordagem focada na otimização e resolução de problemas complexos, integrando os princípios da evolução biológica em conjunto com técnicas e conhecimentos computacionais (FOGEL, 1997). A figura 1 representa a base da CE, mostrando a integração entre conceitos da natureza e da Biologia com técnicas computacionais.

Figura 1 – Relação entre os modelos biológicos e os modelos computacionais



Fonte: Fogel (1997)

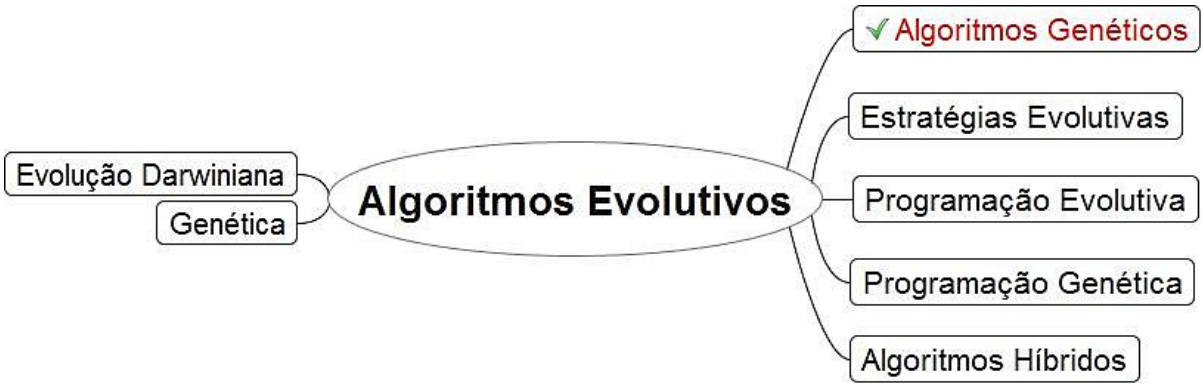
2.1.3 A importância da Computação Evolutiva

A CE é uma subárea dos estudos computacionais inspirada no modelo de evolução natural das espécies escrito por Charles Darwin (1859).

Diferentemente das técnicas e modelos convencionais da computação, a CE se concentra na resolução de problemas através da aplicação de algoritmos heurísticos dedicados à busca e otimização, baseados nos conceitos de evolução e seleção natural da Biologia (AVILA, 2020). É importante ressaltar que os procedimentos heurísticos tratam problemas onde não existe garantia teórica de obtenção de uma solução ótima. A lógica de uma abordagem heurística é garantir que a solução obtida, ao final de um processo, seja a melhor possível quando comparada com outras soluções possíveis.

Segundo Fogel (1997), os Algoritmos Evolutivos podem ser utilizados para qualquer problema ou tarefa de otimização, desde as mais simples às mais complexas, sendo indicados para tarefas heurísticas de busca e otimização devido à sua simplicidade, à sua resposta robusta a mudanças e à sua flexibilidade. A Figura 2 retrata uma visão geral dos Algoritmos Evolutivos, através de conceitos e técnicas que relacionados.

Figura 2 – Visão geral da Computação Evolutiva



Fonte: Silva e Mairink (2019)

A CE compreende uma área muito ampla de técnicas computacionais. Atualmente, as técnicas que compõem a CE são: Algoritmos Genéticos (AGs), Programação Genética (PG), Estratégias Evolutivas (EE), Programação Evolutiva (PE) e os Sistemas de Classificação (SC) (PILA, 2006). A Figura 3 ilustra o conjunto de técnicas que compõe a CE. Cada técnica possui suas peculiaridades e características nas representações dos indivíduos e funcionalidades dos operadores, mas a base e os princípios de todas elas são a teoria evolutiva de Darwin (1859).

As técnicas da CE se inspiram em diversos conceitos evolutivos e da seleção natural para reproduzir funcionalidades computacionais. Como exemplo, nos **Algoritmos Genéticos** (uma das principais técnicas da CE), uma população de indivíduos é criada a fim de reproduzir o ambiente de sobrevivência e seleção natural. Os melhores indivíduos sobrevivem e transmitem suas características a novas gerações. Múltiplas gerações podem

Figura 3 – Composição da Computação Evolutiva

Fonte: Pila (2006)

ser geradas, até que indivíduos com maior grau possível de adaptação ao problema sejam criados.

Possivelmente a maior vantagem dos Algoritmos Evolutivos reside na habilidade de serem mais apropriados à solução de problemas que algoritmos tradicionais não são tão indicados. Embora seja possível utilizar algoritmos convencionais para tentar solucionar qualquer tipo de problema computacional, em muitos casos eles são inadequados para certas categorias de problemas computacionais. A pesquisa em IA evoluiu para uma série de métodos direcionados à solução de problemas específicos em domínios de interesse restrito (FOGEL, 1997).

2.2 Algoritmos Genéticos (AGs)

2.2.1 Perspectiva

A constante evolução da tecnologia transformou a humanidade ao longo da história. Em diversas descobertas e avanços científicos, o ser humano buscou inspiração na natureza. Podemos observar como os pássaros serviram de inspiração direta no desenvolvimento dos aviões. A forma como os girassóis perseguem a luz solar influenciou diretamente a criação das placas solares. O funcionamento das câmeras e lentes fotográficas foi inspirado no sistema ocular (PACHECO, 1999). A quantidade de invenções humanas inspiradas em elementos, formas e fenômenos da natureza é incontável.

Na tentativa de aperfeiçoar técnicas já existentes e encontrar melhores modelos computacionais para solução de problemas, a Ciência da Computação também se beneficiou encontrando inspiração em outras ciências humanas e na própria natureza. A CE surgiu

como um conjunto de paradigmas direcionados à solução de problemas computacionais inspirados no Darwinismo. Os avanços dos estudos da CE possibilitaram a origem de ferramentas e modelos de IA cada vez mais sofisticados. Em 1975, John Holland publicou o livro *Adaptation in Natural and Artificial Systems* (HOLLAND, 1975) que foi um marco para a CE. Holland propôs um método de busca e otimização bioinspirado baseado nos princípios da evolução, seleção natural e genética como recombinação e mutação, conhecido como Algoritmo Genético ou AG.

Os Algoritmos Genéticos são algoritmos iterativos que através de uma amostragem inicial de dados, compostos por um conjunto de possíveis soluções para um determinado problema, produzem futuras gerações (método análogo à reprodução) a cada iteração. Cada um dos indivíduos filhos é gerado a partir da combinação das características de indivíduos pais previamente selecionados (método análogo à seleção natural) de gerações passadas. Assim, os AGs são considerados Algoritmos Evolutivos, por combinar elementos da genética e evolução natural com a computação, fazendo parte tanto da CE, uma subárea da IA.

As soluções, denominadas de indivíduos ou cromossomos, são representadas através de *strings* que podem ser representadas por dados binários, caracteres, números reais ou qualquer tipo de dados que se encaixe no contexto e características do problema. A simulação do processo de evolução se dá por meio da ação de operadores genéticos, os quais realizam variações nos genes, modificando as características dos indivíduos a fim de obter um conjunto de melhores soluções a cada geração (PINTO, 2022).

Métodos heurísticos são algoritmos exploratórios que buscam a solução de problemas. Em Ciência da Computação, o desenvolvimento de um algoritmo busca atender a duas propriedades: ter um tempo de execução aceitável e encontrar a solução ótima, ou provavelmente, aceitável para o problema. Um algoritmo heurístico, entretanto, não necessariamente atende a estas propriedades, de forma que não se pode garantir que ele consiga encontrar alguma solução e nem que a encontrará em tempo hábil (BUENO, 2009).

Os AGs são algoritmos de busca heurísticos não determinísticos e probabilísticos. Por não serem determinísticos, os Algoritmos Genéticos produzem resultados diferentes a cada execução, mesmo utilizando os mesmos parâmetros e valores de entrada. Eles também são considerados probabilísticos porque uma mesma população dificilmente apresentará os mesmos resultados para um mesmo problema (BUENO, 2009).

Para entender os Algoritmos Genéticos é importante conhecer os termos e palavras-chave relacionados a eles. Os AGs possuem um jargão específico de termos, sendo a maioria palavras e termos técnicos encontrados na Biologia e Genética, porém o significado específico de cada termo pertence ao contexto dos Algoritmos Genéticos. A Tabela 1 apresenta os termos mais comuns e seus respectivos significados dentro do contexto dos AGs.

Tabela 1 – Termos e seus respectivos significados nos Algoritmos Genéticos (AG)s

Biologia/Genética	**Algoritmos Genéticos**
Cromossomo	indivíduo, cromossomo, solução
Gene	características do problema
Alelo	valor do gene (característica)
Genótipo	estrutura - conjunto de genes de um indivíduo
Fenótipo	estrutura submetida ao problema

Fonte: Pacheco (1999)

2.2.2 Indivíduo e População

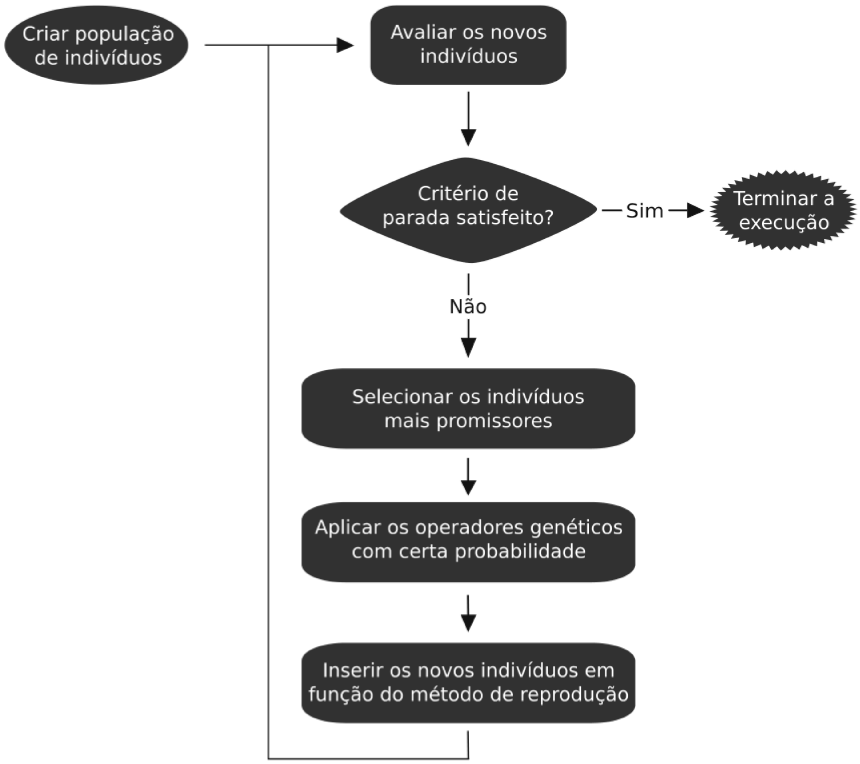
A função de um Algoritmo Genético é pesquisar uma solução ótima dentro do espaço de busca de um problema. Para esse fim, o AG trabalha com um conjunto de soluções candidatas à melhor solução do problema. Cada uma das possíveis soluções do problema é denominada **indivíduo** ou **cromossomo**. O conjunto de soluções candidatas (indivíduos ou cromossomos) é denominado **população**.

De acordo com as características, restrições e natureza do problema, os indivíduos da população inicial podem ser gerados de forma aleatória ou tendenciosa. Na maioria dos casos, a população inicial é criada aleatoriamente (MITCHELL, 1996). A geração aleatória da população inicial proporciona diversidade ao espaço de busca. Geralmente, problemas de simples convergência não são impactados pela alta diversidade; pelo contrário, normalmente são beneficiados, evitando homogeneidade entre os indivíduos e melhorando a convergência. Contudo, em alguns casos, o excesso de diversidade pode afastar a busca das soluções ótimas. Isso geralmente ocorre em problemas complexos com baixo grau de convergência. Nesse tipo de situação, a população inicial pode ser gerada de forma estratégica, baseando-se em algum conhecimento prévio ou heurístico do problema (MITCHELL, 1996). Nesses casos, a população inicial é gerada de forma tendenciosa com o intuito de melhorar a convergência do AG.

Durante a execução de um Algoritmo Genético, populações subsequentes à população inicial são criadas. A Figura 4 representa o ciclo das populações de um Algoritmo Genético.

Diferente dos indivíduos da população inicial que podem ser criados aleatoriamente respeitando regras e restrições do escopo do problema, os indivíduos de populações subsequentes são gerados a partir de operadores genéticos. De maneira similar à genética natural, indivíduos da população inicial, denominados **pais**, são selecionados. Estes indivíduos terão suas informações combinadas, produzindo indivíduos **filhos**. A partir desse ponto, uma nova geração de indivíduos é produzida utilizando uma estratégia de reinserção (DAVIS, 1996). Este processo é realizado por **operadores genéticos** e se repete com o objetivo de se aproximar de uma solução satisfatória a cada nova população gerada.

Figura 4 – Ciclo da geração de populações em Algoritmos Genéticos



Fonte: Ertel (2017)

2.2.3 Função de Avaliação (Fitness Function)

No contexto biológico, uma população é composta por vários indivíduos com variabilidade genética entre si. Os indivíduos mais adaptados ao meio ambiente tendem a sobreviver e se perpetuar, repassando seu material genético para as futuras gerações. Essa teoria é conhecida como *Seleção Natural* (DARWIN, 1859).

Os Algoritmos Genéticos se espelham na biologia natural das espécies. O processo de seleção de indivíduos pais para geração de indivíduos filhos é inspirado na Seleção Natural. No contexto dos Algoritmos Genéticos, o indivíduo representa uma solução candidata à solução ótima do problema. Seguindo essa lógica, cada indivíduo é avaliado, sendo atribuída uma nota (*score*) de desempenho. Esse *score* é denominado aptidão (*fitness*) e mede o quão próximo um indivíduo está de ser a solução do problema.

Na seleção de pais para o *crossover*, indivíduos que representam boas soluções são indivíduos mais próximos da solução ótima e, portanto, possuem maior capacidade de contribuir na busca da melhor solução. Consequentemente, esses indivíduos devem possuir maior probabilidade de serem escolhidos em relação a outros indivíduos. A probabilidade de um indivíduo ser selecionado está relacionada ao seu valor de aptidão (*fitness*). Indivíduos que representam soluções mais próximas à solução ótima possuem maior valor de aptidão, possuindo prioridade e maior chance de serem escolhidos. É importante notar

que indivíduos com baixo valor de aptidão também possuem probabilidade de serem escolhidos. Isso se justifica pelo fato de que mesmo não representando boas soluções quanto os indivíduos com melhores valores de aptidão, eles agregam variabilidade e diversidade no processo de geração de filhos, o que pode ser determinante para encontrar a melhor solução (DAVIS, 1996).

O cálculo do valor de aptidão é feito através de uma função de avaliação ou *fitness function*. A função de aptidão é diferente para cada problema, sendo sua implementação parte crucial no projeto de implementação de um Algoritmo Genético. Uma função de avaliação mal projetada produz resultados inesperados e compromete a convergência de um AG. Da mesma forma, uma função de aptidão bem implementada ao problema a qual ela avalia, tende a favorecer a convergência do AG (DAVIS, 1996). Outra questão importante a respeito da função de avaliação é o seu tempo de processamento. Estimativas demonstram que a maior parte do tempo da execução de um AG está no processamento da função de avaliação (ARORA, 2024). Portanto, a função de avaliação é um dos pontos críticos e requer uma grande atenção na sua implementação.

2.2.4 Operadores Genéticos

Os operadores genéticos provêm os mecanismos necessários para a geração de novas populações de soluções candidatas e manutenção da diversidade dentro do espaço de busca do AG. A busca por uma solução ótima através da geração de sucessivas populações, baseada na combinação das características dos seus indivíduos, representa o conceito-chave dos Algoritmos Genéticos.

Assim como a função de aptidão (*fitness function*), a implementação dos operadores genéticos tem a capacidade de impactar positivamente ou negativamente a convergência de um AG. Uma configuração inadequada dos operadores genéticos pode levar à convergência prematura ou a um esforço computacional elevado que pode inviabilizar a aplicação do AG (PINTO, 2022).

Os Algoritmos Genéticos possuem forte inspiração na biologia evolutiva e na genética natural. Da mesma forma, a funcionalidade dos operadores genéticos é baseada em vários conceitos biogenéticos, como a reprodução, a seleção natural, as alterações genéticas, etc.

Os principais operadores genéticos são: seleção de pais, recombinação (*crossover*), mutação e reinserção (SASTRY, 2005).

A seleção de pais é um preparativo para outro operador genético, o *crossover*. A seleção de pais seleciona indivíduos dentro de uma população para a combinação de suas informações (material genético) durante o *crossover*. Existem vários métodos de seleção, alguns mais restritivos do que outros, assim, a escolha do método de seleção deve considerar as características do problema, visto que impacta diretamente a capacidade de convergência do AG (LIM, 2017).

A função do operador genético recombinação ou *crossover* é gerar indivíduos filhos através da combinação do material genético de indivíduos pais. A ideia de funcionamento do processo é simples: através de um ponto de corte, parte das informações de cada um dos indivíduos pais é mesclada, produzindo indivíduos filhos que combinam suas características (SPEARS, 1993).

Outro importante operador genético é a mutação. A mutação realiza pequenos ajustes no genoma dos indivíduos através da alteração gênica. Genes são aleatoriamente selecionados e têm seus valores modificados com o objetivo de preservar a diversidade genética dentro da população (LIM, 2017).

A reinserção funciona como método para escolha dos indivíduos que farão parte da próxima geração. Existem diversos métodos de reinserção, e a escolha do método mais apropriado irá depender das características do problema (MAN, 1996).

2.2.4.1 Seleção de Pais

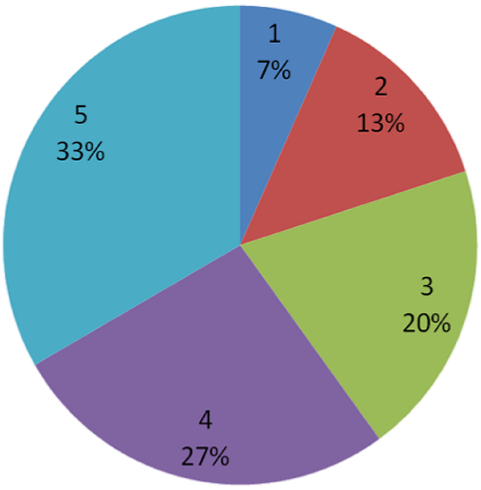
A seleção de pais é um importante processo do ciclo de um AG, pois nesta etapa são selecionados indivíduos para o *crossover* (reprodução). Todos os indivíduos possuem probabilidade de participar e serem escolhidos na seleção de pais para o *crossover*. Porém, de forma similar à seleção natural, os indivíduos mais aptos têm maior probabilidade de serem escolhidos e repassar suas características para a próxima geração. O grau de adaptação de um indivíduo é calculado pela função de avaliação. A seleção dos indivíduos é feita por um método de seleção de pais.

O principal e mais utilizado método de seleção de pais é a **roleta** (*Roulete Wheel*) (PINTO, 2022). O método Roleta foi originalmente escrito por John Holland (1975) em conjunto com os Algoritmos Genéticos na sua obra *Adaptation in Natural and Artificial Systems*. Existem vários métodos de seleção além da roleta, os mais conhecidos são: torneio estocástico (*Stochastic Tournament*), torneio simples (*Simple Tournament*), truncamento (*Truncation*) e ranking (*Rank Based Fitness Assigment*), porém, por ser o método mais tradicional e popular, este trabalho abordará o método de seleção roleta.

Assim como o nome sugere, no método de seleção roleta, os indivíduos são sorteados como se fizessem parte de uma roleta. Todos os indivíduos participam do sorteio, porém, cada indivíduo possui uma probabilidade diferente de ser sorteado. A cada indivíduo é atribuída uma fatia de tamanho diferente da roleta, sendo a sua probabilidade de sorteio diretamente baseada no seu *score* de aptidão, calculado pela função de aptidão. Indivíduos com maior valor de aptidão terão maior probabilidade de serem escolhidos do que os indivíduos com menor valor de aptidão (FILLITO, 2008). A Figura 5 exemplifica este processo de maneira simplificada.

Após as proporções de cada indivíduo forem definidas, a roleta é girada n vezes, sendo n o número de indivíduos a serem escolhidos para o *crossover* (geralmente os Algoritmos Genéticos utilizam somente dois indivíduos pais para o *crossover*). A cada

Figura 5 – Exemplo de sorteio por roleta



Indivíduo	Fitness	Probabilidade
1	1.0	7%
2	2.0	13%
3	3.0	20%
4	4.0	27%
5	5.0	33%

Fonte: elaborado pelo autor

vez que a roleta for girada, um indivíduo é selecionado. É importante observar que após um indivíduo ser selecionado, ele não pode participar do próximo sorteio da roleta para evitar a possibilidade de que ele seja sorteado novamente e haja duplicidade de pais no *crossover*.

Os indivíduos selecionados para a roleta são chamados de pais, porque terão seus genes combinados, formando indivíduos filhos.

2.2.5 Recombinação (*crossover*)

O operador genético *recombinação* ou *crossover* é baseado nos mecanismos de reprodução da natureza. Os cromossomos são compostos por milhares de genes, estruturas básicas responsáveis pela hereditariedade (SCHUBERT, 2007). Durante o processo de reprodução sexual dos seres vivos, dois pais têm as informações de seus cromossomos combinadas para formar novos indivíduos (filhos). Os filhos são formados pela mistura dos genes dos dois pais. De forma congênere, no *crossover*, dois cromossomos pais (soluções candidatas escolhidas por um método de seleção de pais) produzem novas soluções filhas através da combinação dos genes dos pais. Os indivíduos filhos são compostos por parte das características de ambas as soluções pais, sendo cada um dos filhos uma nova solução candidata à solução ótima.

A reprodução resulta em novas combinações genéticas, o que aumenta a diversidade e pode levar à evolução da espécie. O mesmo ocorre nos Algoritmos Genéticos. O *crossover* possibilita o surgimento de novas soluções e, consequentemente, soluções potencialmente melhores. Porém, genes ruins também podem ser repassados aos filhos, resultando em soluções filhas piores. Eventualmente, tais soluções possuirão valores de aptidão baixos,

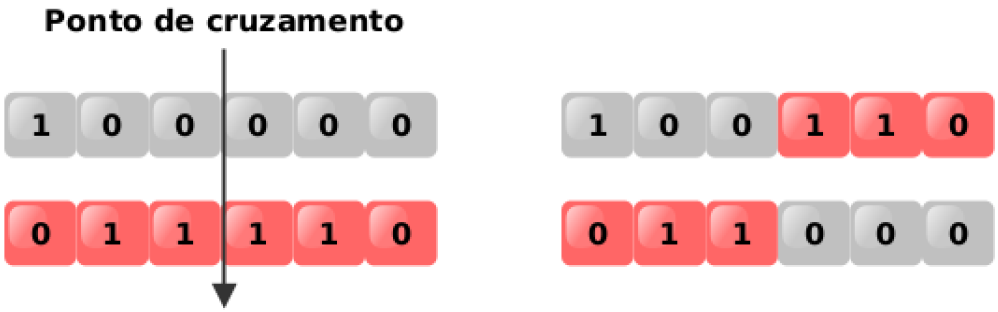
dificultando que sejam escolhidas para participar das próximas populações a serem geradas.

Para que os indivíduos filhos herdem os genes dos pais, é necessário estabelecer quais genes de cada pai serão herdados em cada filho. Existem várias estratégias diferentes para isso, sendo os principais e mais comuns tipos de *crossover*: o *crossover* simples e o *crossover* múltiplo.

O *crossover* simples é o tipo de *crossover* mais utilizado e simples de se implementar. No *crossover* simples, dois pais geram dois filhos, sendo um ponto de cruzamento (também chamado de ponto de corte) definido aleatoriamente. A função do ponto de cruzamento é separar a sequência de genes de cada pai em duas partes. A partir desse momento, uma parte do genoma de cada pai é mesclada formando um genoma completo (FILLITO, 2008).

Na Figura 6 temos dois indivíduos pais e seus respectivos genótipos representados em binário, sendo o genótipo do pai1 100000 e o genótipo do pai2 011110. O ponto de cruzamento define onde inicia e onde termina o segmento de bits que irá compor os dois indivíduos filhos. O genótipo do filho1 é definido pelos bits 100 (herança do pai1 - primeira metade do seu ponto de cruzamento) e 110 (herança do pai2 - segunda parte do seu ponto de cruzamento), enquanto o genótipo do segundo filho é composto pelos segmentos 011 (herança do pai2 - primeira parte do seu ponto de cruzamento) e 000 (herança do pai1 - segunda parte do seu ponto de cruzamento).

Figura 6 – Crossover simples - ponto de corte e geração de filhos

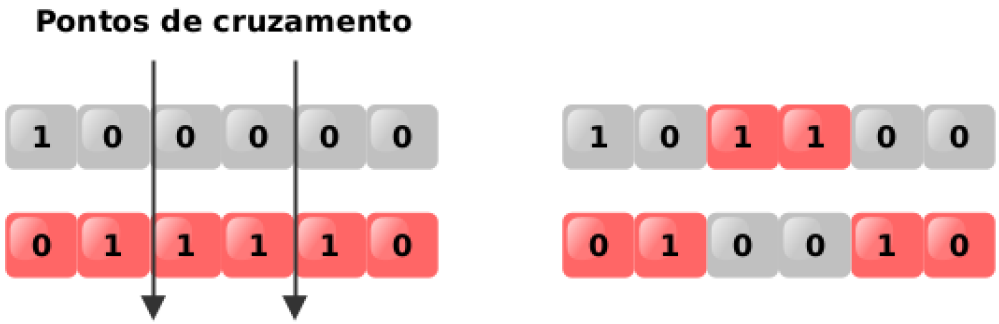


Fonte: elaborado pelo autor

O *crossover* múltiplo funciona de maneira parecida com o *crossover* simples, porém, a diferença é que no *crossover* múltiplo são definidos vários pontos de cruzamento. Da mesma forma como o *crossover* simples, os segmentos de genes são combinados, produzindo os indivíduos filhos. A Figura 7 ilustra um exemplo de *crossover* múltiplo. Podemos observar que há dois pontos de cruzamento, que fazem com que o genoma dos pais seja dividido em três sequências de genes ao invés de apenas duas sequências de genes, como acontece no *crossover* simples. O filho1 possui duas sequências de genes do pai1 e uma

sequência de genes do pai2, enquanto que o filho2 ocorre o inverso (duas sequências de genes do pai2 e apenas uma sequência de genes do pai1).

Figura 7 – Crossover múltiplo com dois pontos de corte



Fonte: elaborado pelo autor

2.2.6 Mutação

A evolução representa a origem de toda biodiversidade na natureza. Na Genética, os cromossomos são as unidades portadoras da herança nuclear de todas as células através do DNA (SCHUBERT, 2007). A mutação é responsável pela geração de variabilidade genética dentro da população de seres vivos da mesma espécie através de alterações no DNA.

A variabilidade genética em populações de indivíduos proporciona novas características que podem ser herdadas pelos descendentes desses indivíduos. Estas novas características permitem a geração de indivíduos mais adaptados ou menos adaptados que os indivíduos já existentes. Porém, a seleção natural favorece que os indivíduos mais adaptados sobrevivam e perpetuem seu material genético nas futuras gerações de descendentes (AZEVEDO, 2000).

O melhoramento genético de indivíduos de uma espécie acontece através da mutação. A evolução das espécies acontece mediante melhoramento genético e da seleção natural. Assim, a mutação é considerada uma das bases da evolução (WOODRUFF; THOMPSON, 1998).

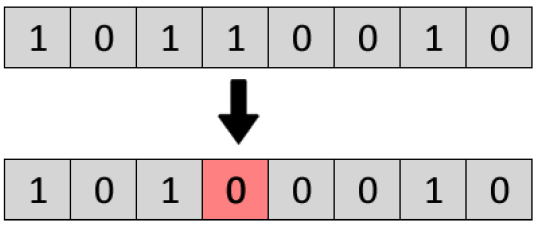
O operador genético mutação, como o próprio nome sugere, funciona de maneira similar às mutações ocorridas no DNA. Dentro do contexto dos AGs, a mutação realiza pequenas alterações no genoma de determinados cromossomos de uma população. Aleatoriamente, um ou mais genes são sorteados e seus valores são levemente modificados. O objetivo é explorar novas regiões do espaço de busca de soluções para o problema através da introdução de diversidade genética à população.

A mutação ocorre após o *crossover*. Contudo, a probabilidade de aplicação da mutação ocorre mediante o valor da taxa de mutação. A taxa de mutação é um parâmetro que estabelece a frequência com que a mutação é aplicada nos cromossomos. Quanto maior seu valor, maior a probabilidade de um cromossomo ser escolhido para sofrer mutação.

A mutação depende do tipo de dados dos genes. Genes podem ser representados por valores binários, números, caracteres, etc. Os tipos mais comuns de mutação são: mutação binária, mutação real e a permutação.

Como o próprio nome diz, a mutação binária ocorre em cromossomos representados por valores binários. Um bit é selecionado aleatoriamente e o seu valor é alternado. Se o valor do bit escolhido é zero, ele é alterado para um, e vice-versa. A Figura 8 exemplifica a mutação binária. Nesse exemplo, o quarto bit foi selecionado. O seu valor que era um foi alterado para zero.

Figura 8 – Mutação Binária



Fonte: elaborado pelo autor

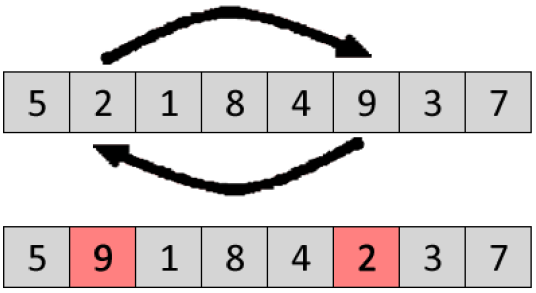
Na mutação real, o valor do gene sofre uma pequena modificação. O valor original do gene sorteado é incrementado ou decrementado. O tipo da operação de modificação, (incremento ou decremento) é determinado por sorteio. O valor a ser incrementado ou decrementado também pode ser determinado por sorteio dentro de um range de valores ou pode ser preestabelecido durante a inicialização do AG. A Figura 9 ilustra um exemplo de mutação real.

Figura 9 – Mutação Real

Fonte: elaborado pelo autor

Já a permutação é uma modalidade de mutação onde há o intercâmbio de valores entre dois genes. A figura 10 demonstra um exemplo de permutação. Neste exemplo, o segundo e o sexto gene foram escolhidos para permutação. Ao final da mutação, seus valores são permutados.

Figura 10 – Mutação - Permutação



Fonte: elaborado pelo autor

2.2.7 Reinserção

Após os processos de *crossover* e mutação, um conjunto de indivíduos filhos é gerado através de uma população de indivíduos pais. A partir desse ponto, uma nova população de indivíduos é criada através de uma seleção entre os indivíduos existentes. O operador genético *reinserção* define a estratégia de seleção utilizada na composição da nova população. Na reinserção, os indivíduos existentes (pais e filhos) competem para fazer parte da nova população. Existem vários métodos (estratégias) de reinserção; as principais são: reinserção pura, reinserção uniforme, melhores pais e filhos e o elitismo.

No método reinserção pura, a nova população é composta exclusivamente pelos indivíduos filhos gerados. Já no método reinserção uniforme, a seleção de indivíduos é feita através de um sorteio entre os indivíduos pais e os indivíduos filhos. Assim como no operador genético *seleção de pais*, a reinserção uniforme utiliza a roleta ou o torneio estocástico para a seleção de indivíduos da nova população.

Os métodos de reinserção mais comuns são *melhores pais e filhos* e o *elitismo*. O objetivo do método melhores pais e filhos é escolher os melhores indivíduos para compor a próxima geração, independentemente de serem indivíduos pais ou indivíduos filhos. Um Algoritmo Genético que trabalha com uma população de n indivíduos, os indivíduos pais e filhos são colocados juntos, e os n melhores indivíduos irão compor a nova população. A escolha dos melhores indivíduos é feita pelo valor de aptidão. Os n indivíduos com maior valor de aptidão são os escolhidos (MITCHELL, 1996).

No método elitismo, um certo número de indivíduos com melhor aptidão são selecionados. Estes indivíduos são denominados *elite*. O número de indivíduos escolhidos para compor a elite é determinado pelo parâmetro taxa de elitismo. Os indivíduos selecionados

como elite são adicionados diretamente à nova população sem passar pelo cruzamento ou mutação. Isso garante que as melhores soluções já encontradas se perpetuem na próxima geração. As vagas restantes da nova população são preenchidas por indivíduos gerados pelos operadores genéticos padrão (seleção de pais, *crossover* e mutação), ou seja, pelos indivíduos filhos gerados pelo processo padrão (MAN, 1996).

2.2.8 Critérios de Parada

Os critérios de parada são as condições que determinam quando o Algoritmo Genético deverá se encerrar. Pinto, Martarelli e Nagano (2022), definem os critérios de Parada como as condições que determinam a finalização do Algoritmo Genético, adotando o indivíduo com maior aptidão como a solução do problema. Esses critérios são muito importantes e precisam ser objetivos e bem definidos, pois é necessário garantir o uso eficiente dos recursos computacionais (como memória e processamento que são finitos), além de evitar a finalização precoce do algoritmo.

A definição dos critérios de parada deve considerar a natureza e peculiaridades do projeto, visto a diversidade de condições e heterogeneidade de características que projetos diferentes podem apresentar. Portanto, há uma grande margem de possibilidades de critérios de parada.

A **descoberta de uma solução ótima** talvez seja o critério de parada mais óbvio e comum. Algoritmos Genéticos não garantem o descobrimento de uma solução que satisfaça os requisitos do projeto, porém encontrar uma solução ótima é o objetivo principal do uso de um Algoritmo Genético. Assim, ao encontrar a solução definitiva para o problema ou uma solução satisfatória as necessidades do problema, o AG encontra o seu objetivo, encerrando a sua execução.

Outro critério de parada bastante comum é o **número de gerações**. Problemas com alta dificuldade de otimização e alta dificuldade (ou até impossibilidade) de descoberta de uma solução satisfatória, independentemente das circunstâncias, podem ser executados infinitamente. Para evitar a execução infindável e desperdício de recursos computacionais, o número máximo de gerações pode ser definido como critério de parada. Assim, ao atingir o número de gerações predefinido na inicialização, mesmo que uma solução ótima não tenha sido encontrada, o AG é finalizado.

O **tempo limite de execução** é outro critério de parada frequentemente utilizado. Através da definição desse critério na inicialização, o AG é executado por um período fixo e previamente determinado de tempo. Após atingir esse tempo de execução, o AG é encerrado.

Em casos onde o AG tenha dificuldade em convergir encontrando a solução exata com 100% de valor de aptidão ou situações que não seja possível encontrar tal solução, é plausível definir um **valor de aptidão alvo** como critério de parada. Ao definir um valor

de aptidão-alvo, o AG considerará uma solução como solução ótima quando encontrar uma solução com o valor definido ou superior ao valor-alvo e será finalizado.

Além dos critérios mencionados, é possível definir **critérios específicos**, definidos com base nas características e particularidades do domínio do problema (DAVIS, 1996).

Por fim, podem ser definidos **critérios compostos** de parada. Um critério composto é uma combinação de dois ou mais critérios de parada. Assim, o AG só é encerrado após a satisfação simultânea dos critérios contidos no critério composto (DAVIS, 1996).

2.2.9 Parâmetros do AG

Os resultados da execução de um Algoritmo Genético dependem de vários fatores, entre eles, os parâmetros definidos na inicialização. Diversas características e comportamentos de um AG são influenciados por parâmetros (MAN, 1996). A definição dos parâmetros deve considerar as particularidades do problema e ser feita com diligência, pois eles influenciam o desempenho, o uso de memória e a capacidade de convergência (capacidade de se encontrar a solução ótima) (MITCHELL, 1996). Os principais parâmetros definidos em um AG são: tamanho das populações, taxa de mutação, número de gerações, taxa de incremento ou decremento e o valor de aptidão alvo.

A quantidade de indivíduos que compõe a população em um AG é ajustada via parâmetros. Pode-se ajustar tanto o tamanho da população inicial quanto o tamanho das populações subsequentes. O **tamanho da população** é um dos parâmetros com maior impacto no desempenho de um AG. Populações muito grandes consomem muitos recursos computacionais. Por outro lado, grandes populações cobrem um vasto espaço de busca do problema. Assim, é importante estabelecer o tamanho da população considerando tanto o desempenho quanto o espaço de busca de soluções (PINTO, 2022).

A **taxa de mutação** define o percentual de indivíduos que serão selecionados para sofrer mutação após o *crossover*. Uma taxa alta aumenta a diversidade de indivíduos dentro da população, porém altas taxas de mutação podem tornar quase aleatória a busca por soluções, perdendo uma das essências do AG, a manutenção das características dos indivíduos mais aptos nos indivíduos descendentes.

O parâmetro **número de gerações** define o número máximo de ciclos populacionais que o AG produzirá. Esse parâmetro é utilizado como critério de parada de um AG para situações em que o AG não consegue encontrar uma solução ótima após várias iterações.

O parâmetro **taxa de incremento ou decremento** determina o valor a ser somado ou subtraído ao valor do gene sorteado durante uma operação de mutação. É importante observar que esse parâmetro só faz sentido existir se o problema a ser tratado pelo AG necessitar do uso de mutação real. Em casos de mutação binária ou mutação por permutação, esse parâmetro é desnecessário.

O **valor de aptidão alvo** também é outro parâmetro opcional. Este parâmetro define um valor de aptidão (*fitness value*) específico, que quando obtido ou ultrapassado, indica

que o AG encontrou uma solução satisfatória e, portanto, a execução do AG pode ser finalizada.

Além dos parâmetros citados, outros parâmetros de inicialização podem ser definidos de acordo com a natureza e peculiaridades do problema. Além disso, variações do Algoritmo Genético convencional podem possuir operadores genéticos adicionais ou exigir critérios ou regras específicas que necessitem da definição de parâmetros adicionais (MAN, 1996).

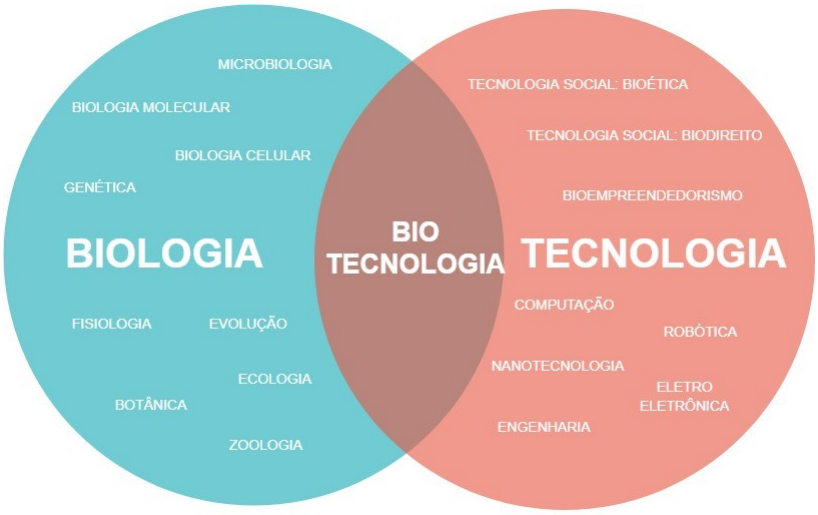
2.3 Algoritmo Genético Transgênico

2.3.1 Transgenia e os Organismos Geneticamente Modificados

Na década de 50, Watson e Crick (1953) descobriram o modelo de estrutura em dupla hélice da molécula de DNA. O DNA é composto por duas fitas de nucleotídeos em espiral, interligadas por ligações de hidrogênio entre as bases nitrogenadas. Essa descoberta foi extremamente relevante, pois foi a base para a compreensão da estrutura do material genético, da transmissão hereditária e do sequenciamento do genoma. A descoberta da molécula de DNA mudou a ciência, permitiu importantes progressos na Genética, o avanço da era moderna da Biologia, além do surgimento de novas ciências a partir dos desdobramentos desta descoberta.

A Biotecnologia foi uma das novas ciências derivadas dos avanços da Biologia Molecular. A Biotecnologia é uma ciência multidisciplinar que compreende o conjunto de técnicas de Engenharia Genética que utilizam seres vivos (ou parte dos seres vivos), no desenvolvimento de processos e produtos a serviço de interesses econômicos e/ou sociais (ALBERTS, 2002). A Figura 11 apresenta as subáreas que compõem a Biotecnologia.

Figura 11 – Diferentes áreas científicas que compõem a Biotecnologia



Fonte: BQIBR (2017)

As técnicas de Engenharia Genética utilizam a tecnologia do DNA recombinante. A tecnologia do DNA recombinante é caracterizada pelo processo de manipulação do material genético de organismos vivos, através da introdução ou alteração direta de sequências específicas de DNA. Um gene estrategicamente escolhido é isolado do DNA de um organismo. Através do uso de um vetor (molécula de DNA capaz de transportar o gene de interesse para dentro de uma célula hospedeira) o gene é introduzido em uma célula hospedeira de um organismo de destino, modificando suas propriedades e estruturas. A célula hospedeira se multiplica, fazendo cópias do DNA recombinante (WATSON, 2014). A figura 12 ilustra o processo.

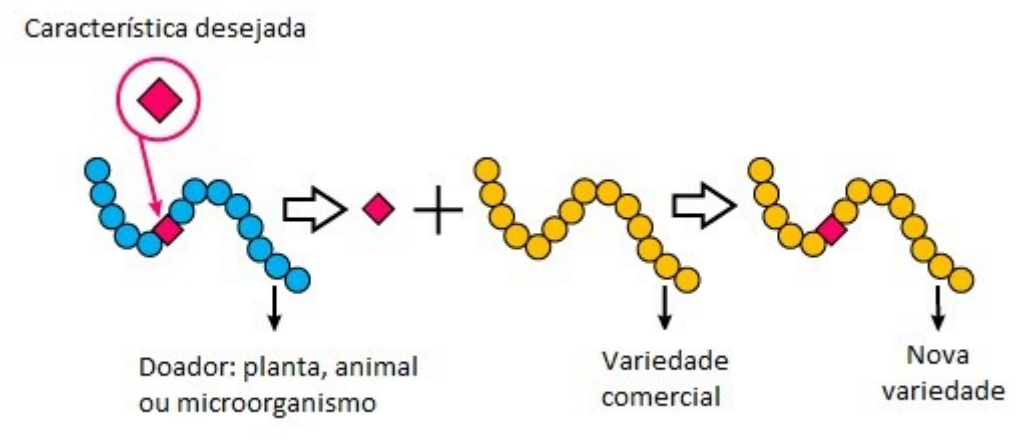


Figura 12 – Processo de Transgenia: Tecnologia do DNA Recombinante

Fonte: manualdabiologia (2024)

A tecnologia do DNA recombinante transpõe as barreiras de cruzamento entre os reinos animal, vegetal, protista e fungi. O resultado do processo é um indivíduo geneticamente igual ao indivíduo hospedeiro (indivíduo que recebeu a molécula de DNA recombinante), porém acrescido de nova(s) característica(s) genética(s), proveniente(s) do outro indivíduo (AZEVEDO, 2000). Esses indivíduos são conhecidos como Organismos Geneticamente Modificados ou transgênicos. Um transgênico é um Organismo Geneticamente Modificado (OGM), porém nem todo OGM é obrigatoriamente um transgênico; organismos geneticamente modificados são todos organismos que tiveram seu material genético alterado por meio de técnicas de engenharia genética, enquanto que os transgênicos são indivíduos que tiveram seu material genético modificado pela introdução de parte do material genético de outro organismo (VARGAS, 2018).

O melhoramento genético de plantas, animais e demais organismos vivos só é possível se existir variabilidade genética, isto é, se, dentro das espécies a serem melhoradas, existirem indivíduos com uma vasta quantidade de diferentes características que possam ser herdadas. Essa variabilidade tem origem, fundamentalmente, em mutações, que são a base da evolução. A transgenia advém da recombinação, resultante da hibridação, ou seja, do cruzamento genético entre indivíduos portadores de diferentes mutações, promovendo

variabilidade adicional a populações de indivíduos (XAVIER, 2008). A transgenia tem o poder de adicionar características que naturalmente não são encontradas em um grupo de organismos, resultando no melhoramento genético e aumento da diversidade gênica dentro de uma população.

Atualmente, a transgenia faz parte de diversos processos produtivos, como na produção de medicamentos, desenvolvimento de vacinas, diagnóstico de doenças, melhoramento de ingredientes naturais, conservação ambiental, entre outros, sendo amplamente utilizada na indústria farmacêutica, indústria de cosméticos, na medicina, agricultura e produção de alimentos. A agricultura e a produção de alimentos talvez sejam os melhores casos de sucesso do desenvolvimento de transgênicos. A introdução dos OGMs na agricultura possibilitou o aumento da produtividade agrícola, redução de custos de produção e trouxe mais segurança na produção de alimentos ao permitir o cultivo de variedades de culturas mais resistentes a condições climáticas diversas.

Um bom exemplo de transgênico na agricultura é a soja transgênica. A soja transgênica é um tipo de soja modificada geneticamente para ser resistente ao herbicida glifosato; diferente da soja convencional que não é resistente a esse herbicida. A soja transgênica é resistente a insetos e à seca, auxilia o controle de plantas daninhas (sem afetar a soja) sendo mais produtiva e lucrativa aos agricultores (AZEVEDO, 2000). Outro bom exemplo é o milho Bt. O milho Bt é o milho transgênico, modificado geneticamente pela inclusão dos genes da bactéria *Bacillus thuringiensis* (Bt). A tecnologia Bt adiciona resistência a insetos e pragas, removendo a necessidade de pulverização de inseticidas em lavouras, diminuindo os custos de produção e, ao mesmo tempo, melhorando a qualidade da colheita (AZEVEDO, 2000).

Por fim, mas não menos importante, o uso de transgênicos e OGMs pode resolver problemas ambientais, reduzindo os custos com a recuperação de ecossistemas, e consequentemente, melhorar a qualidade de vida das populações locais. Além disso, o uso de transgênicos no meio ambiente abre oportunidades para difusão das indústrias verdes e inovações para tecnologias ambientais.

2.3.2 Variações do Algoritmo Genético

A seção 2.2 descreveu detalhadamente o fluxo de funcionamento do Algoritmo Genético convencional. Este padrão foi definido por Holland (1975), e desde então, é amplamente utilizado no contexto de problemas de busca e otimização.

Contudo, uma das características dos Algoritmos Genéticos é a sua adaptabilidade às características do problema que se propõe satisfazer. Durante décadas, novos operadores genéticos e inclusive novos modelos completos de Algoritmos Genéticos surgiram para se adaptar a características e restrições de problemas específicos em que o AG convencional seria menos efetivo.

Desse modo, assim como a própria capacidade de adaptabilidade dos seres vivos, que através da evolução, buscam se aprimorar e se adequarem às condições que são expostos, novos modelos e propostas de Algoritmos Genéticos continuam surgindo como novas alternativas de resolução de problemas e desafios complexos, que exigem a formulação de técnicas mais adaptadas às suas características. Como exemplo, há variações do AG convencional que incorporaram diferentes conceitos provenientes da genética como dominância, translocação, diferenciação sexual, deleção, *crossover* haplóide, mutacão, diploidia, transgenia entre outros (GOLDBERG, 1989). O **Algoritmo Genético Transgênico** é uma das variações do Algoritmo Genético convencional baseado nas técnicas da engenharia genética, mais especificamente nas alterações gênicas por introdução de genes relevantes.

2.3.3 Operador Transgenic

A Biotecnologia é uma ciência moderna e promissora, que combina conhecimentos biogenéticos com técnicas de diferentes áreas, como engenharia, química e tecnologia na elaboração de soluções inovadoras. Uma das maiores inovações provenientes da Biotecnologia são os organismos geneticamente modificados (OGMs).

Nos últimos anos, o uso de organismos geneticamente modificados tem crescido exponencialmente, tanto na agricultura quanto em diversos outros setores produtivos da sociedade. A grande razão para isso deve-se ao fato de que os OGMs são produzidos por meio de processos de melhoramento genético (técnicas de DNA recombinante), apresentando características desejáveis superiores às características dos organismos não-transgênicos. Talvez a maior vantagem dos transgênicos é a capacidade de adicionarem características genéticas naturalmente inexistentes aos organismos não-transgênicos provenientes de outras espécies de organismos.

Com base em todas as vantagens observadas na transgenia, Amaral e Hruschka (2013) propuseram um novo modelo de variação do Algoritmo Genético convencional, através da formulação de um novo operador genético; o **operador transgênico**. O operador transgênico induz a modificação de genes desejáveis em cromossomos com baixo potencial (indivíduos com os menores valores de aptidão dentro de uma população).

O operador transgênico é fortemente inspirado na transgenia. A principal motivação do operador transgênico é a produtividade do AG, acelerar o tempo de descoberta da solução ótima; tanto pela perspectiva do tempo de execução quanto pela perspectiva do número menor de gerações necessárias para encontrar a solução.

O operador transgênico é visto como uma estratégia de elitismo dirigida focada em genes alvos específicos: o conjunto de genes/valores identificados como relevantes para o surgimento de comportamentos e características que aproximem a solução candidata a solução ótima. Ao identificar os genes/valores mais relevantes dentro de uma população,

o operador transgênico aplica estes valores nos respectivos genes de indivíduos com baixo valor de aptidão previamente selecionados.

Capítulo 3

Trabalhos Correlatos

Algoritmo Genético é um tema popular e frequentemente discorrido em trabalhos de pesquisa científica no campo dos algoritmos evolucionários. Inúmeros tópicos sobre Algoritmos Genéticos são abordados em pesquisas sobre o tema, como: desempenho, acurácia, variações (novas técnicas ou operadores genéticos), parametrização, problemas e dificuldades de otimização, etc. Além dos diferentes tópicos sobre AGs discutidos em estudos científicos, vários trabalhos abordam o uso de AGs em diferentes contextos e para diferentes fins. O uso de Algoritmos Genéticos na otimização de fórmulas matemáticas é um exemplo de um contexto do uso de AGs diretamente relacionado ao tema principal deste estudo.

Analisando os trabalhos da literatura, este capítulo apresenta uma seleção de trabalhos considerados pertinentes e relevantes aos temas centrais desta pesquisa científica. Este capítulo apresentará, discutirá e destacará os principais elementos destes trabalhos, os quais, contribuíram na fundamentação e construção de cada etapa desta pesquisa.

3.1 Exploring Optimization Technique: Genetic Algorithm, 2024

Este artigo trata de várias técnicas de otimização para problemas matemáticos, desde métodos tradicionais até abordagens mais sofisticadas. As principais técnicas citadas pelo artigo são: algoritmos evolucionários, métodos do gradiente, programação linear, programação não linear, programação dinâmica, recozimento simulado e heurística. Uma parte de destaque são os Algoritmos Genéticos. O artigo delineia os princípios básicos dos Algoritmos Genéticos, abordando os operadores genéticos (seleção, recombinação e mutação), o processo de avaliação de aptidão e trazendo exemplos de códigos. Adicionalmente, o artigo apresenta avanços recentes no campo, incluindo algoritmos multiobjetivo e híbridos.

De um modo geral, o ponto de destaque apresentado por Arora, Sahlot e Bala (2024), é a forma como ele aborda os conceitos dos Algoritmos Genéticos e demais técnicas de

otimização na solução de problemas matemáticos. Além disso, a pesquisa cita como algoritmos evolucionários e demais técnicas de otimização podem ser utilizadas na solução de problemas do mundo real.

3.2 A Real Coded Genetic Algorithm For Solving Integer And Mixed Integer Optimization Problems, 2009

Deep, Singh, Kansal e Mohan (2009) propõe uma variação de Algoritmo Genético para lidar com problemas lineares e não-lineares de otimização com restrições de números inteiros e inteiros mistos, denominado MI-LXPM, com base em outra variação de Algoritmo Genético, o LXPM. O MI-LXPM implementa um procedimento específico de truncamento para lidar com restrições de números inteiros em variáveis de decisão com uso de uma abordagem de penalidade sem parâmetros no tratamento de restrições ocorridas em problemas de otimização.

Um ponto de destaque desta pesquisa, realçado nos seus experimentos, foi a dificuldade de acerto na definição dos parâmetros corretos para um Algoritmo Genético. A dificuldade no ajuste da parametrização adequada dificultou muito a condução do experimento e a obtenção de resultados satisfatórios. Somente após a identificação da configuração apropriada dos parâmetros, é que Deep, Singh, Kansal e Mohan (2009) obtiveram resultados esperados e desejáveis.

3.3 Transgenic, An Operator For Evolutionary Algorithms, 2011

O artigo trata do operador genético *transgenic*, um novo operador genético baseado nos benefícios encontrados nos OGMs através da transgênese. A tecnologia do DNA recombinante, muito utilizada em processos biotecnológicos, permite isolar, manipular e combinar sequências de DNA, estrategicamente selecionadas, de diferentes indivíduos com a intenção de adicionar características dos indivíduos doadores nos indivíduos receptores. O operador *transgenic* foi projetado para reproduzir os mecanismos de ação da transgenia nos Algoritmos Genéticos.

Os experimentos realizados nesta pesquisa envolveram o uso do operador transgênico para trabalhar com ontologia genética e dados sintéticos. Os resultados dos experimentos demonstraram que o operador transgênico foi prolífico, sendo muito eficaz em aumentar o poder de convergência do AG nos testes realizados, sendo indicado para cenários com restrição no número de gerações. A capacidade do operador *transgenic* em acelerar a con-

vergência, diminuindo o número de gerações criadas, além de apresentar bons resultados com tarefas de classificação, demonstrou que o operador *transgenic* é uma abordagem bastante promissora para Algoritmos Genéticos.

3.4 Transgenic: An Evolutionary Algorithm Operator, 2013

O artigo *Transgenic: an evolutionary algorithm operator*, também é um artigo de Amaral e Hruschka sobre Algoritmos Genéticos e o operador *transgenic*. O estudo optou por utilizar uma configuração básica para o operador Transgênico com o intuito de demonstrar o potencial do operador na sua forma mais simples.

Durante os testes realizados nesta pesquisa, foi constatado que o operador *transgenic* exercia uma forte pressão seletiva sobre os datasets utilizados, fazendo com que os indivíduos mais aptos (maior *fitness*) tivessem uma grande vantagem de perpetuar seus genes na reprodução (*crossover*). A vantagem desse cenário é facilitar que o algoritmo convirja rapidamente. Porém, pode ocasionar problemas de estacionamento em ótimos locais, o que exigiria o uso de taxas de mutação mais altas para explorar o espaço de busca de forma mais ampla do que as taxas convencionais de mutação.

Desse modo, a pesquisa demonstrou que o operador transgênico é bastante promissor, mesmo em sua forma mais básica. Contudo, a alta pressão seletiva pode exigir cuidados com a taxa de mutação. O artigo sugere que pesquisas futuras devem ser realizadas para explorar o potencial do operador em diferentes contextos e configurações, a fim de compreender mais o seu potencial e aprimorar ainda mais seu desempenho e aplicabilidade.

3.5 Binary or Integer Chromosome: Which Is the Best Structure for Supervised Machine Learning Using Genetic Algorithms?, 2025

No artigo *Binary or Integer Chromosome: Which Is the Best Structure for Supervised Machine Learning Using Genetic Algorithms* os autores Alves, Neto, Gomes, Santos, Bertarini e Amaral (2025) buscam produzir um gerador de conjunto de regras de classificação simples e interpretáveis com alta precisão, capaz de gerar valores precisos e melhores taxas de convergência. A pesquisa busca também realizar uma verificação sobre os efeitos da representação binária analisando as taxas de convergência e precisão. Para alcançar estes objetivos, são utilizados três geradores de conjunto de regras, o Computational Evolutionary Environment (CEE), o Non-linear Computation Evolutionary Environment (NLCEE) e o Binary Nonlinear Computational Evolutionary Environment (BIN-NLCEE).

O BIN-NLCEE é na verdade, um refinamento do NLCEE idealizado por Amaral e Hruschka (2010). O BIN-NLCEE possui uma alteração na estrutura cromossômica, utilizando uma representação binária em sua estrutura cromossômica no intuito de simplificar sua mutação e também gerar um maior espaço de busca. O BIN-NLCEE possui o potencial de refinar o processo de busca ao permitir mutações simples "*bit-flip*", que impactam significativamente a exploração do espaço de solução e ajudam a evitar mínimos locais. Essa habilidade melhora a robustez do algoritmo, mantendo a eficiência computacional. Além disso, a codificação binária simplificou a interpretabilidade e a implementação do método, tornando-o mais acessível e adaptável a diferentes tarefas de classificação.

Durante a pesquisa, cinco conjuntos de dados (*datasets*) são utilizados, todos relacionados à área médica: breast-w, Pima-Indians-diabetes, Parkinson, mammography mass e COVID-19. O *dataset* COVID-19 foi criado pelos autores através de dados públicos compartilhados pelo Grupo Fleury no Brasil. Os demais *datasets* foram retirados do repositório online de acesso público da *UCI Machine Learning Repository*. Além dos *datasets*, foram utilizados quatro classificadores tradicionais: J48, IBK, Bayes naive e SVM.

Os resultados dos experimentos realizados apresentaram a obtenção de bons valores de aptidão nos três métodos (CEE, NLCEE e o BIN-NLCEE). Contudo, de um modo geral, o BIN-NLCEE apresentou melhores resultados e conseguiu atingir menor variância quando comparado com todos os classificadores tradicionais atingindo resultados mais consistentes.

Uma observação importante de se mencionar é que o operador transgênico de Amaral e Hruschka (2011) não é utilizado nessa pesquisa. Contudo, ao lado de outros modelos de AGs que podem ser utilizados para auxiliar métodos de classificação, o AG Transgênico é citado, além de ser descrito o seu mecanismo de funcionamento.

3.6 IG-CEE: An Embedded Information Gain approach to Genetic Algorithms, 2021

A classificação é uma das tarefas mais importantes na área de Mineração de Dados, sendo objeto de estudo de grande interesse atualmente nas pesquisas de IA. Diversos estudos estão surgindo sobre o uso de AGs na geração de regras de classificação *IF-THEN* para classificação de dados. Amaral e Hruschka (2010) desenvolveram um AG capaz de gerar regras *IF-THEN* simples e compreensíveis através do banco de dados da Gene Ontology (GO), denominado CEE. Buscando resultados mais eficientes, Alves, Mendes, Gomes, Bertarini e Amaral (2021) propuseram um método inspirado no CEE denominado Information Gain-based Computational Evolutionary Environment (IG-CEE).

O IG-CEE utiliza a abordagem *Information Gain (IG)* para selecionar o melhor conjunto de atributos que irão compor as regras *IF-THEN*. O IG é uma medida de avaliação de atributos, utilizada para definir a probabilidade que cada atributo terá para aparecer

na regra. O objetivo da abordagem IG é produzir melhores taxas de classificação através da seleção do melhor subconjunto de atributos para compor as regras *IF-THEN*.

Durante a pesquisa, o operador *transgenic* de Amaral e Hruschka (2011) foi utilizado em um Classificador baseado em um AG. Se um valor específico S de um gene G apresentar uma forte influência no valor de aptidão, o operador *transgenic* força alguns indivíduos a herdarem o valor S no mesmo gene G.

Foram utilizados três *datasets* durante os experimentos contendo 3 classes, 1.200 amostras e 20 atributos no total, nomeados *D1*, *D2* e *D3*. Comparando os resultados do CEE e do IG-CEE, o CEE não obteve um desempenho satisfatório em selecionar corretamente os atributos relevantes. Apenas a classe 2 no *dataset D1* e um único atributo na classe 1 no *dataset D3* foram selecionados corretamente. O IG-CEE por outro lado, conseguiu selecionar os atributos mais relevantes para compor as regras *IF-THEN* em quase todos os cenários, falhando apenas na classe 2 do *dataset D2* e na classe 3 do *datsaset D3*. Em 36 comparações realizadas entre o IG-CEE e os classificadores convencionais, o IG-CEE obteve os melhores resultados em 97,23% dos casos (em 35 comparações). De um modo geral, a pesquisa demonstrou que o IG-CEE foi capaz de selecionar os melhores atributos para compor as regras *IF-THEN*, além de alcançar melhores valores de aptidão, gerando regras com maiores taxas de classificação.

3.7 The Effects of "Preferentialism" on a Genetic Algorithm Population over Elitism and Regular Development in a Binary F6 Fitness Function, 2016

Os Algoritmos Genéticos surgiram como uma abordagem de solução de problemas computacionais baseados nos conceitos evolutivos (MITCHELL, 1996). Nos AGs cada indivíduo é um cromossomo particular, sendo parte integrante de uma população de diversos indivíduos. O *crossover* é um operador genético que pode ou não ser aplicado em todos os indivíduos de uma população, sendo a chance de um indivíduo ser selecionado para o *crossover* definida pelo seu valor de aptidão (MAN, 1996). O preferencialismo é uma abordagem que garante que pelo menos um indivíduo especial seja selecionado. O artigo de Boeira (2016) realiza um estudo analítico entre dois modelos de preferencialismo: o preferencialismo de mutação e o segundo melhor preferencialismo.

Durante os experimentos, duas funções de AGs foram utilizadas, a função *Binary F6* e uma função de sequenciamento. A função *Binary F6*, define cromossomos com 44 genes binários. A função busca converter todos os genes para zero, resultando em uma aptidão de 1. A função *Binary F6* foi analisada da seguinte forma: considerou-se o número médio de iterações necessárias para cada método (sem preferência, elitismo, preferência por mutação e segunda melhor preferência) alcançar um valor de aptidão de 0,995, bem como

a aptidão média da população final com cromossomo de aptidão 0,995. Para os casos em que o algoritmo não atingiu a aptidão de 0,995 dentro de 100.000 iterações, esse limite fixo foi utilizado para análise. Adicionalmente, foi realizada uma comparação entre a melhor aptidão obtida e a média de aptidão da população. A segunda função utilizada se baseia em um algoritmo de sequenciamento com taxas variáveis de mutação e cromossomos compostos por uma sequência ordenada de números inteiros (REEVES, 1995). O objetivo é determinar o melhor número de *crossovers* preferidos (REEVES, 1995). A análise da função de sequenciamento espera a obtenção de uma aptidão máxima de zero. Caso esse objetivo não seja alcançado dentro de 3 milhões de iterações, a execução é interrompida e os resultados parciais são exibidos. A fim de determinar as configurações ideais para o cruzamento de gerações subsequentes, foram avaliadas três estratégias: ausência de preferência, elitismo e preferência por mutação, utilizando os seguintes pontos de cruzamento: 3, 5, 8, 12 e 20.

Os resultados dos experimentos demonstraram que gerações de curto prazo podem apresentar resultados insatisfatórios em relação a seus descendentes e na qualidade dos genes que eles transmitem. Além disso, estender demasiadamente a vida útil das gerações não justifica o uso do elitismo, pelo contrário, o torna desnecessário. A função de sequenciamento e a função *Binary F6* demonstraram o preferencialismo é certamente melhor do que um sistema sem preferencialismo. Outra conclusão que os resultados evidenciam é que o preferencialismo alcança valores de aptidão mais elevados com menos iterações em comparação com métodos que não utilizam preferencialismo. Além disso, ao analisar a variância ou a taxa de aptidão por iteração, o preferencialismo se destaca em relação ao elitismo. Como um processo natural em sistemas reprodutivos, o preferencialismo também demonstrou superioridade como método computacional. A pesquisa exposta por este artigo não faz uso do operador *transgenic*, porém, a pesquisa cita variações de Algoritmos Genéticos e cita o operador transgênico como uma nova abordagem relacionada à concepção de transgenia.

3.8 A New Frequency Analysis Operator for Population Improvement in Genetic Algorithms to Solve the Job Shop Scheduling Problem, 2021

Os problemas de otimização combinatória (Combinatorial Optimization Problems (COPS)) envolvem a busca da configuração mais vantajosa de parâmetros, por meio da análise de diferentes permutações de elementos pertencentes a um conjunto finito. Devido à sua ampla aplicabilidade, esses problemas têm sido amplamente explorados por pesquisadores em diversos domínios e contextos. O Job-Shop Schedulig Problem (JSSP) é um tipo de problema de otimização combinatória, sendo tema frequente de pesquisas acadê-

micas. O JSSP busca processar um conjunto finito de trabalhos utilizando um número também finito de máquinas. O principal objetivo nesse tipo de problema é definir uma sequência de processamento das tarefas que otimize determinados critérios, como, por exemplo, a minimização do tempo total de utilização dos recursos disponíveis. O artigo de Viana, Contreras e Junior (2022) propõe a utilização de um operador de orientação com a função de modificar indivíduos com baixa adaptação por meio da incorporação de material genético proveniente de indivíduos altamente adaptados. Além disso, introduz-se um novo método para avaliar a qualidade genética dos indivíduos, baseado na análise de frequência genética. A abordagem proposta é testada em uma variedade de algoritmos genéticos contemporâneos, considerando dois estudos de caso definidos por *benchmarks* consagrados do problema de escalonamento de tarefas JSSP presentes na literatura. O JSSP representa um desafio significativo do ponto de vista computacional, uma vez que desenvolver métodos capazes de encontrar soluções exatas com bom desempenho, dentro de um tempo de processamento aceitável, não é uma tarefa trivial, mesmo para instâncias de pequeno ou médio porte. Diante dessa complexidade, implementações de algoritmos cada vez mais eficazes em fornecer soluções aproximadas em tempos computacionalmente viáveis surgem para lidar com JSSP.

Durante a pesquisa, foi desenvolvido um operador de melhoramento genético baseado na análise da frequência de genes de indivíduos bem adaptados da população, com alta adaptabilidade de integração a qualquer tipo de AG denominado de GIFA. O GIFA foi testado em 58 diferentes instâncias conhecidas de JSSP de diferentes complexidades, sendo integrado a cinco diferentes metaheurísticas baseadas em AGs. Com o intuito de analisar diversos aspectos, cada uma das técnicas, tanto em sua forma original quanto combinada com o GIFA, foi executada 35 vezes. Os resultados permitiram observar que a presença do GIFA contribuiu para tornar as metaheurísticas mais consistentes, uma vez que houve redução na média e no desvio padrão dos resultados. Adicionalmente, na maior parte dos casos, observou-se uma melhoria nos piores resultados encontrados por cada técnica durante as execuções. Um comportamento semelhante foi identificado nas melhores soluções encontradas por cada método. Os resultados fortalecem a premissa de que o uso do GIFA favorece as metaheurísticas baseadas em Algoritmos Genéticos, contribuindo para a descoberta de soluções melhores. Por fim, há um planejamento para a exploração de técnicas de aprendizado profundo, com ênfase em métodos de aprendizado por reforço, visando a identificação de influências genéticas em cromossomos dentro de populações de algoritmos genéticos e a ampliação de aplicação do operador GIFA, estendendo-o a outros problemas que envolvem otimização combinatória, como os problemas de pseudocoloração em grafos.

Como proposta para trabalhos futuros, planejamos explorar técnicas de aprendizado profundo, com ênfase em métodos de aprendizado por reforço, visando identificar influências genéticas em cromossomos dentro de populações de algoritmos genéticos. Também

temos a intenção de ampliar a aplicação do material desenvolvido, estendendo-o a outros problemas dentro do mesmo domínio, como o Flexible Job Shop Scheduling, bem como a diferentes categorias de desafios que envolvem otimização combinatória, como os problemas de pseudocoloração em grafos.

3.9 Application Programming Interface Library para Algoritmo Genético com Operador Transgênico, 2013

O objetivo principal da pesquisa de Rodrigues, Souza, Aranha e Freitas (2013), foi realizar a validação de uma Application Programming Interface Library (API) de um Algoritmo Genético utilizando o operador transgênico de Amaral e Hruschka (2013). A validação da API é feita através de mineração de dados em um *dataset* com dados sobre tráfego de carros. Com base nos dados presentes no *dataset*, uma solução ótima é identificada como uma regra IF-THEN que indique com maior precisão a existência ou não de congestionamentos.

A API foi executada 100 vezes, apresentando convergência do AG com uso do operador transgênico. Com base nos dados do *dataset* utilizado, o AG encontrou as seguintes regras que apresentaram valor de aptidão de 100%:

- IF *Fluxo* $>= 51$ THEN *Congestionado*
- IF *Fluxo* < 50 THEN *Não Congestionado*

A API utilizou taxa de mutação de 2% e o método de reinserção melhores pais e filhos. Uma parte determinante para o sucesso de todo AG durante o projeto de implementação é a definição da *Fitness Function* (DAVIS, 1996). Uma *Fitness Function* bem implementada pode reduzir o tempo de processamento e aumentar a capacidade de convergência (ARORA, 2024). O cálculo da *Fitness Function* utilizada pela API foi definido por: $TP \times 0.5 + FP \times 0 * 5$, sendo TP os registros classificados corretamente e FP os registros que foram ditos como não pertencentes à classe avaliada.

Na conclusão final, a pesquisa foi considerada bem-sucedida por conseguir identificar soluções ótimas (regras que identificaram situações de congestionamento e não congestionamento do trânsito). Contudo, o trabalho é lacônico por infelizmente não apresentar nada sobre a implementação do operador transgênico. O uso do operador transgênico é apenas citado durante o texto, mas não é demonstrado ou explicado nada sobre o seu funcionamento e nem sua implementação na API. O texto apenas menciona que o AG utiliza o operador transgênico de Amaral e Hruschka (2013), mas nada além disso. O grande mérito do trabalho e sua contribuição a esta pesquisa está no relato das etapas

de validação de um AG transgênico e na descrição de funcionalidades específicas de um AG convencional, como *crossover*, mutação, reinserção, cálculo da *Fitness Function*, que foram de grande relevância para a convergência da API.

3.10 Transgenic Genetic Algorithm to Minimize the Makespan in the Job Shop Scheduling Problem, 2020

O artigo *Transgenic Genetic Algorithm to Minimize the Makespan in the Job Shop Scheduling Problem* de Viana, Junior e Contreras (2020) propõe a aplicação do operador *transgenic* de Amaral e Hruschka (2011) no problema de escalonamento de JSSP.

O JSSP é um problema computacional de otimização combinatória considerado difícil por demandar restrição de precedência entre máquinas e um grande espaço de busca combinatória (GEN, 1994). O JSSP é tema de estudo frequentemente encontrado na literatura relacionado à classe de problemas NP-Difíceis. O algoritmo convencional usado para resolver o problema é o método *branch-and-bound*. De acordo com Gen, Tsujimura e Kubota (1994), o JSSP é um tipo de problema que possui N trabalhos a serem processados em M máquinas com o seguinte contexto:

1. Nenhuma máquina pode processar mais de um trabalho por vez;
2. Nenhum trabalho pode ser processado por mais de uma máquina por vez;
3. A sequência de máquinas que um trabalho visita é completamente especificada e tem uma estrutura de precedência linear;
4. Os tempos de processamento são conhecidos;
5. Todos os trabalhos devem ser processados em cada máquina apenas uma vez.

O problema de JSSP consiste em agendar um conjunto de trabalhos (*jobs*) em um conjunto de máquinas, de forma a minimizar o tempo total de processamento (*makespan*) ou outro critério de desempenho. O objetivo principal é encontrar um cronograma que minimize o tempo total decorrido. O método *branch-and-bound* para resolver JSSP geralmente produz bons resultados, porém, geralmente consome um alto tempo computacional. Desse modo, Viana, Junior e Contreras (2020) propuseram um estudo sobre uma alternativa ao método *branch-and-bound*, utilizando o Algoritmo Genético Transgênico.

Uma importante contribuição desta pesquisa é propor e elaborar uma versão alternativa do operador *transgenic* idealizado para reduzir o *makespan* em problemas de *job shop scheduling*. O operador transgênico utilizado na pesquisa é inspirado no operador transgênico proposto por Amaral e Hruschka (2011), porém adaptado para proporcionar

melhores resultados na otimização do JSSP. A proposta comparou o Algoritmo Genético Transgênico com outras abordagens de resolução do JSSP descritas em trabalhos relacionados, utilizando como critério de avaliação o tempo para obtenção da resposta e a redução do valor de *makespan*. O Algoritmo Genético Transgênico foi comparado com Ant Colony Optimization (ACO) (KATO, 2009), Adaptive AG (MORANDIN, 2008) e o AG proposto por Morandin, Kato, Deriz e Sanches (2008). Estatisticamente, o Algoritmo Genético Transgênico apresentou resultados superiores às demais técnicas avaliadas. Contudo, trabalhos futuros são necessários para avaliar sua eficiência em problemas similares ao JSSP.

3.11 A Cellular Automata-Based Path-Planning for a Cooperative and Decentralized Team of Robots, 2019

A coordenação de movimentos Multi-Robot Systems (MRS) possui várias aplicações úteis, que vão desde a execução de missões de busca e salvamento, vigilância até mapeamento colaborativo. O planejamento de trajetórias seguras que evitem colisões é fundamental para robôs autônomos, principalmente quando estejam operando em ambientes desconhecidos. Essa tarefa se torna ainda mais desafiadora em sistemas com múltiplos robôs, especialmente quando é necessário que todos ajustem seus percursos a fim de preservar a formação original do grupo. As estratégias utilizadas para coordenação de atividades com grupos de múltiplos robôs é desafiadora. Dentre as abordagens de coordenação, o controle de formação destaca-se como uma das mais simples. Diversas técnicas têm sido propostas para lidar com esse desafio. Contudo, a navegação de robôs em tempo real exige, frequentemente, soluções eficientes e com baixo consumo de recursos computacionais. Entretanto, muitos dos métodos existentes ainda apresentam um custo computacional elevado. O estudo de Oliveira, Silva, Ferreira, Couceiro, Amaral, Vargas e Martins (2019), propõe e realiza a avaliação de um modelo descentralizado e autônomo voltado ao planejamento de trajetórias para equipes de robôs baseado em Cellular Automata (CA) (autômatos celulares) para resolver um problema de planejamento de trajetória e controle de formação usando regras locais e estados discretos. O modelo baseado em CA definido na pesquisa utiliza regras de decisões locais para definir o próximo movimento, mas baseado apenas na vizinhança dos robôs e na comunicação com outros robôs próximos.

O Algoritmo Genético Transgênico é citado no artigo, quando o artigo debate estratégias de regras para CA. Várias regras CA são inspiradas *neighborhood states*, os quais, se guiam pela leitura de sensores locais. Os *neighborhood states* de cada robô são estipulados por regras de classificação IF-THEN. As regras de CA para operações de desvios são muito sensíveis e possuem dificuldade na classificação precisa das células vizinhas como

células livres ou de obstáculo. Dessa forma, o artigo sugere o operador transgênico como uma das abordagens eficientes capazes de serem utilizadas em Algoritmos Genéticos para mineração de regras IF-THEN que permitam a classificação precisa de células vizinhas.

Foram conduzidas simulações e experimentos com robôs reais em cenários envolvendo um e três agentes. Os resultados demonstraram que, mesmo com recursos computacionais limitados, foi possível otimizar tanto as trajetórias quanto o tempo de execução das tarefas. Desse modo, os resultados indicam que o modelo proposto é viável para aplicação em robôs com hardware modesto, como baixa capacidade de memória e processadores de desempenho reduzido. No modelo atual, a formação da equipe permanece fixa, e o robô designado como coordenador ocupa sempre a posição central na configuração, mantendo essa função mesmo diante de deslocamentos significativos para desviar de obstáculos. Além disso, um novo esquema de classificação de vizinhança foi gerado através de um Algoritmo Genético. As novas regras são capazes de determinar os estados das células adjacentes com base nas leituras dos sensores em ambientes ruidosos. Também foi testado e implementando uma estrutura de equipe com estratégias de comunicação mais descentralizadas. Os resultados das simulações e dos experimentos com robôs reais demonstraram eficácia e robustez. Assim, o modelo CA elaborado na pesquisa possibilitou que robôs atuassem de forma autônoma durante as manobras de desvio de obstáculos, enquanto operaram de maneira coordenada para preservar o padrão de formação do grupo. Graças à introdução de novos estados e ao aprimoramento do conjunto de regras, os robôs demonstraram um desempenho mais eficaz na evasão de obstáculos. De modo geral, as trajetórias geradas apresentaram características mais suaves e naturais, o que contribuiu para a redução dos erros de odometria e, consequentemente, para uma estimativa mais precisa de suas posições atuais. Como perspectiva futura, pretende-se investigar o uso de regras locais baseadas em autômatos celulares para possibilitar o rearranjo dinâmico da estrutura de formação durante a execução das tarefas.

3.12 Multiple Disjunctions Rule Genetic Algorithm (MDRGA): Inferring Non-Linear IF-THEN Rules in Non-Linear Datasets, 2018

Um dos principais objetivos da mineração de dados é a descoberta e extração de conhecimento dentro de grandes conjuntos de dados. O trabalho *Multiple Disjunctions Rule Genetic Algorithm (MDRGA): Inferring Non-Linear IF-THEN Rules in Non-Linear Datasets* de Matos e Amaral (2018) implementa um novo AG denominado Multiple Disjunctions Rule Genetic Algorithm (MDRGA). O MDRGA se propõe a identificar regras de classificação IF-THEN não lineares de alto nível, precisas e preditivas, com alta clareza e confiabilidade a partir de *datasets* não lineares. Essas regras devem conciliar alta

capacidade preditiva com a facilidade de interpretação e compreensão do conhecimento extraído, sendo mais precisas do que os métodos de classificação tradicionais encontrados em AGs.

Para cada classe, o MDRGA produz uma única regra clara e concisa, permitindo disjunções e conjunções da regra antecedente. O conjunto de informações extraídas pelo MDRGA delineia a relevância dos atributos de cada classe, como também a relação entre os atributos mais relevantes. Os resultados dos experimentos com o MDRGA demonstraram diferenças sutis entre os resultados de treinamento e teste. Outra observação encontrada foi que cada regra apresentou um pequeno número de antecedentes na parte IF da regra. O MDRGA se mostrou eficiente na classificação de problemas reais onde a descoberta de conhecimento não linear de alto nível sobre o domínio é desejável. Trabalhos futuros foram idealizados com o objetivo de experimentar o uso do MDRGA em *datasets* não lineares como *Madelon*, *Bupa* e *Ionosphere*. Outra abordagem planejada em trabalhos futuros é aplicar o MDRGA em conjuntos de dados sintéticos lineares e não lineares para verificar profundamente o desempenho do MDRGA em um ambiente controlado. O Algoritmo Genético Transgênico é citado no artigo quando o artigo cita abordagens eficientes e competentes na geração de regras de classificação IF-THEN para AGs.

3.13 Construção de Conhecimento de Alto Nível a Partir de Datasets Biológicos com Alta Dimensionalidade Utilizando Algoritmos Genéticos, 2011

A pesquisa original de Amaral e Oliveira (2010) implementou um Algoritmo Genético especializado na busca de regras de alto nível do tipo IF-THEN com potencial de delimitar possíveis genes relacionados a nove classes de câncer e seus respectivos níveis de expressão, conseguindo assim associações gene/câncer e gene/gene. Os nove tipos de câncer analisados foram: mama, sistema nervoso central, colón, leucemia, melanoma, pulmão, ovário, renal e células reprodutivas. O trabalho *Construção de Conhecimento de Alto Nível a Partir de Datasets Biológicos com Alta Dimensionalidade Utilizando Algoritmos Genéticos* de Silva, Rodrigues, Bevilaqua, Barreto, Nascimento e Amaral (2011) tem a finalidade de demonstrar a capacidade de convergência do AG de Amaral e Oliveira (2010) na mineração de regras em bases de dados biológicas, porém em contexto de alta dimensionalidade. Diferentemente da pesquisa original que trabalha no máximo com 55 genes utilizando métodos propostos por outros autores como método de pré-seleção gênica, o trabalho de Silva, Rodrigues, Bevilaqua, Barreto, Nascimento e Amaral (2011) reformulou o AG de Amaral e Oliveira (2010) para tratar expressões gênicas de 1000 genes utilizando o método de validação *leave-one-out*, considerado como o método mais apropriado para problemas desta natureza, em substituição ao método 2:1 utilizado no estudo original.

A pesquisa utilizou os dados presentes na base de dados NCI60, composta pela expressão de 1.000 genes (colunas), medida para 61 amostras de células (linhas), sendo cada amostra classificada em uma das nove classes de câncer citadas anteriormente. Os experimentos foram executados utilizando os mesmos valores de parâmetros utilizados por Amaral e Oliveira (2010). Contudo, o AG não convergiu com os parâmetros originais. O motivo mais provável seria a alta dimensionalidade (1000 atributos) do problema. Mesmo assim, há a expectativa de que o ambiente evolutivo desenvolvido tenha capacidade de convergir utilizando a base de dados NCI60. Para isso, uma série de mudanças é proposta e planejada para serem aplicadas. Entre as alterações propostas para a convergência do ambiente evolutivo na base de dados NCI60, os autores citam o uso do operador *transgenic* de Amaral e Hruschka (2011). O operador *transgenic* possui a capacidade de aumentar a convergência de AGs através da identificação e replicação de genes relevantes ao aumento do valor de aptidão. Além do uso do operador *transgenic*, outras mudanças foram propostas como o uso de taxas de *crossover* menores do que 100%; uso de taxas de mutação maiores, comparação dos resultados obtidos com outros métodos tradicionais de classificação no ambiente Weka, aplicação dos conjuntos de genes em ferramentas tradicionais de análise de bioinformática como: BLAST, PHYLIP, ClustalW e ALIGN; entre outras medidas.

3.14 Lógica para Semáforo Inteligente Baseado na Mineração de Dados por Algoritmo Genético Transgênico, 2013

O crescimento expressivo no número de automóveis e os problemas derivados do congestionamento impulsionaram o desenvolvimento de estudos voltados à melhoria do sistema de trânsito. Existem várias abordagens para tentar solucionar o problema de fluxo de tráfego. Estudos baseados em modelos matemáticos e o uso de autômatos celulares para controle do tempo do semáforo e coordenação da duração de ciclos dos sinais são encontrados na literatura. A pesquisa de Cunha, Ferreira, Rodrigues e Alencar (2013) propõe uma nova metodologia para a otimização do fluxo de tráfego utilizando heurísticas de Inteligência Artificial. Mais precisamente, a pesquisa propõe o uso de Algoritmos Genéticos com o operador transgênico de Amaral e Hruschka (2011) para automatizar o controle dos semáforos em tempo real. Essa abordagem visa permitir decisões dinâmicas baseadas nas condições de tráfego, com o objetivo de otimizar o fluxo de veículos em áreas críticas e momentos de alta concentração de fluxo de veículos na malha urbana.

O AG Transgênico desenvolvido por Cunha, Ferreira, Rodrigues e Alencar (2013) utiliza mineração de dados para extrair padrões comportamentais e definir um conjunto de regras adaptativas. Tais regras são integradas a um sistema baseado em lógica *Fuzzy*,

o qual automatiza o controle semafórico em tempo real, promovendo a tomada de decisão dinâmica com base nas condições observadas do ambiente viário, de modo a maximizar a fluidez em trechos críticos da malha urbana. A lógica *Fuzzy* trabalha com graus de pertinência no intervalo entre 0 à 1 para criar regras difusas para fazer inferência e alcançar a solução ótima do problema. O principal objetivo deste trabalho é demonstrar a eficácia do Algoritmo Genético Transgênico na tarefa de mineração de dados, e sua eficiência na extração de regras difusas a partir de uma base de dados de tráfego. Além disso, busca-se evidenciar a eficiência da lógica *Fuzzy* no controle semafórico. Como diferencial em relação a trabalhos anteriores, a base de dados utilizada representa o fluxo real de tráfego de um trecho urbano da cidade de Jataí, garantindo que as regras geradas reflitam fielmente o comportamento do tráfego local.

Todos os experimentos foram realizados através de simulações, porém utilizando dados reais de trânsito da cidade de Jataí no trecho da avenida Goiás com a avenida Brasil. Os resultados indicaram que o Algoritmo Genético Transgênico proporcionou um desempenho 50% superior em comparação ao Algoritmo Genético convencional. Outro ponto importante identificado foi que o sistema *Fuzzy* mostrou-se aplicável ao controle semafórico inteligente, permitindo que os semáforos tomem decisões adaptadas às condições do ambiente. Em todos os testes realizados no simulador, o comportamento do semáforo foi positivo. Os autores propõem uma análise mais aprofundada das regras utilizadas, a realização de testes com múltiplos semáforos para avaliar a sincronização entre eles, uma nova coleta de dados com variáveis adicionais para mineração, e, se possível, a implementação do sistema em um semáforo real para verificar seu desempenho prático como propostas de trabalhos futuros.

3.15 Crossover and Mutation Operators of Genetic Algorithms, 2017

Lim, Sultan, Sulaiman, Mustapha e Leong (2017)

Entre os principais operadores genéticos estão o *crossover* e a mutação. Ambos os operadores possuem o potencial de impactar positivamente e negativamente a capacidade de convergência de um AG. É comum que AGs enfrentem problemas relacionados a convergência lenta e à estagnação em regiões de ótimos locais. Isso ocorre, em grande parte, devido à baixa diversidade dentro da população. Os operadores *crossover* e a mutação tem grande influência na evolução populacional, pois possibilitam o aumento na diversidade populacional e a geração de melhores soluções candidatas.

O operador *crossover* combina as informações dos indivíduos da população para explorar o espaço de busca de maneira mais eficiente, contribuindo significativamente para o desempenho do AG. A mutação modifica os genes dos descendentes. Ao introduzir variações na população, a mutação aumenta a diversidade e permite que o algoritmo investigue

novas regiões do espaço de busca, ajudando a evitar que o processo evolutivo fique preso em soluções locais.

Diversos estudos têm sido conduzidos ao longo dos anos para investigar diferentes técnicas de mutação com o objetivo de melhorar o desempenho dos Algoritmos Genéticos (AGs). A função principal da mutação é modificar os genes dos descendentes, promovendo uma maior diversidade dentro da população. Esse aumento na variedade genética ajuda o algoritmo a escapar de soluções locais ou subótimas, prevenindo a convergência antecipada.

O sucesso na aplicação dos AGs está fortemente ligado à eficácia dos operadores de cruzamento e mutação, que são os principais responsáveis por orientar a busca em direção aos ótimos globais — ou seja, identificar os valores máximos ou mínimos globais das funções objetivo. Manter um equilíbrio entre as capacidades de exploração (busca ampla no espaço de soluções) e intensificação (busca refinada em regiões promissoras), representadas por esses dois operadores, é essencial para tornar o processo de busca mais rápido e garantir resultados de alta qualidade.

Capítulo 4

Proposta

Neste capítulo, são exploradas e detalhadas duas implementações de Algoritmos Genéticos especializadas na otimização de fórmulas matemáticas: o **StandardAG** (uma implementação de um Algoritmo Genético convencional) e o **TransgenicAG** (um Algoritmo Genético que utiliza o operador *Transgenic*); para fins deste estudo denominado Algoritmo Genético Transgênico.

A seção 4.1 descreve de maneira geral a proposta e o ciclo de processos que compõem o funcionamento do StandardAG. A seção 4.2 detalha o TransgenicAG, desde os conceitos envolvidos no seu projeto, até a sua implementação; explicando de maneira clara e objetiva o funcionamento do operador genético *transgenic*. Por fim, a seção 4.3 apresenta as equações matemáticas escolhidas para serem otimizadas pelas duas propostas de AGs durante os experimentos.

4.1 StandardAG

4.1.1 Visão geral

Para a realização deste estudo, foi proposta a criação de dois Algoritmos Genéticos direcionados à otimização de fórmulas matemáticas; um Algoritmo Genético convencional e um Algoritmo Genético Transgênico (Algoritmo Genético com operador *transgenic*). Ambos algoritmos foram implementado utilizando a linguagem de programação Java.

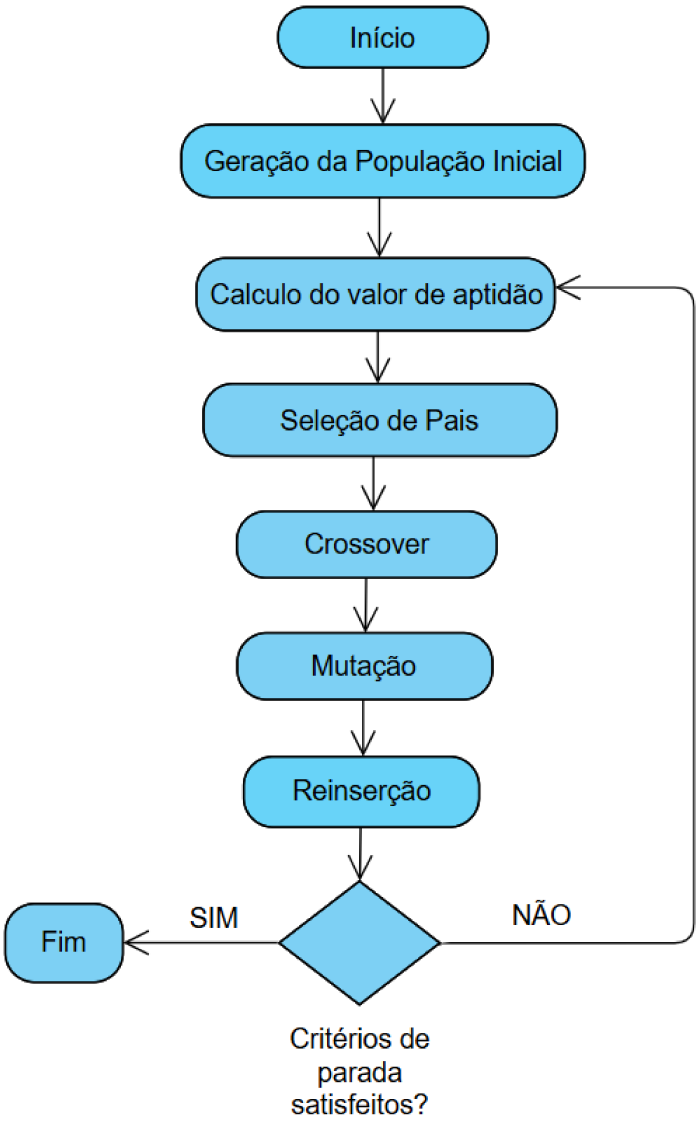
O primeiro algoritmo proposto e implementado para a pesquisa é o **StandardAG**; um Algoritmo Genético convencional, porém especificamente projetado para otimização de fórmulas matemáticas. A Figura 13 ilustra o fluxo natural de um Algoritmo Genético convencional, citando os operadores genéticos em sua respectiva ordem de execução. O algoritmo 1 representa o fluxo de operações do StandardAG. Conforme pode ser observado, o StandardAG segue o mesmo fluxo de funcionamento e operações de um AG convencional.

O StandardAG foi desenvolvido com o objetivo de otimizar a busca por mínimos ou máximos globais em fórmulas matemáticas. O Código 1 exibe o código-fonte da classe

principal (arquivo Main.java) do StandardAG.

Todo o fluxo de operações do StandardAG, descrito pelo Algoritmo 1, é executado na chamada do método *executar()* da classe "AG". O método recebe um objeto que implementa a interface *MathFunction* (arquivo MathFunction.java). A interface *MathFunction* representa uma abstração para fórmulas matemáticas no programa. As fórmulas matemáticas em si são codificadas por classes que implementam a interface *MathFunction*.

Figura 13 – Fluxo de processos realizados pelo Algoritmo Genético convencional



Fonte: elaborado pelo autor

Algoritmo 1 – Fluxo geral do Algoritmo Genético convencional

1: **procedure** StandardAG ▷ fluxo geral do StandardAG
2: CriarPopulacaoInicial()
3: $continuidade \leftarrow verificaContinuidade()$
4: **while** $continuidade$ = TRUE **do**
5: SelecionarPaisCrossover()
6: CrossOver()
7: $m \leftarrow verificaProbabilidadeMutacao()$
8: **if** m = TRUE **then**
9: Mutacao()
10: **end if**
11: Reinsercao()
12: $continuidade \leftarrow verificaContinuidade()$
13: **end while**
14: **end procedure**

Código 1 – AG Classe Main.java

```
//Algoritmo Genetico em Java.
//Inicio do codigo da classe principal (Classe Main.java)
package br.ufu;

import br.ufu.mathfunctions.Beale;
import br.ufu.mathfunctions.*;

public class Main {

    public static void main(String[] args) {

        long inicio = System.currentTimeMillis();
        AG ag = new AG();
        ag.executar(new Beale());
        long fim =System.currentTimeMillis();
        System.out.println("Fim da execucao: "
        + (fim-inicio) + milissegundos");
    }
}
// Fim do programa
```

O programa calcula o tempo de execução total do AG (em milissegundos) e o exibe em uma mensagem no final da execução. Além dessa mensagem, outras mensagens são

exibidas durante a execução. Quando o StandardAG encontra a solução ótima, o programa exibe uma mensagem informando o valor da solução ótima e o número de gerações criadas pelo AG até encontrar a solução. Da mesma forma, quando o AG atinge o número máximo de gerações definido via parâmetro na inicialização, ele exibe uma mensagem informando que atingiu o número máximo de gerações e não encontrou a solução. Ambas as mensagens são geradas pelo método *verificaContinuidade()* da classe AG. Este método, como o próprio nome sugere, gerencia a continuidade e o encerramento do AG através dos critérios de parada. Foram definidos apenas dois critérios de parada: a descoberta de uma solução ótima e o número máximo de gerações.

O método *executar()* da classe "AG"reproduz o fluxo geral de etapas realizadas pelo AG de maneira semelhante à Figura 13. O Código 2 retrata o método *executar()*.

Código 2 – AG Método *executar*() (Classe AG.java)

```
//O metodo geraPopulacaoFilhos() serve para gerar cromossomos filhos
    derivados dos cromossomos da geracao atual de individuos
//Os operadores geneticos: selecao pais, crossover e mutacao sao
    implementados por metodos invocados dentro do metodo
    geraPopulacaoFilhos();
public int executar(MathFunction funcaoMatematica) {
    this.funcaoMatematica=funcaoMatematica;
    if(this.seed==null)
        this.r = new Random();
    else
        this.r = new Random(this.seed);

    inicializaPopulacao();
    verificaContinuidade(this.populacao);
    while(this.continuidade){
        geraPopulacaoFilhos();
        reinsercao();
        verificaContinuidade(this.populacao);
    }
    return this.indexPopulacao;
}

public void definirSeed(long seed){
    this.seed=seed;
}
```

O método *inicializaPopulacao()* cria a população inicial do Algoritmo Genético. Tanto a quantidade da população inicial quanto a quantidade de indivíduos das populações subsequentes é definida via parâmetro durante a inicialização do AG. Os parâmetros são definidos na seção 5.2.3. Os indivíduos da população inicial são criados aleatoriamente; cada função matemática possui um intervalo de valores de entrada específicos. Para cada

gene dos cromossomos criados na população inicial, é atribuído um valor dentro deste intervalo de valores definido por cada função. Na seção 4.3, são mencionadas as funções matemáticas e seus respectivos intervalos de valores de entrada. Após a criação de cada indivíduo, é calculado o seu valor de aptidão.

O método *geraPopulacaoFilhos()* é responsável por criar novos indivíduos que poderão compor a nova geração populacional. A seleção dos indivíduos que farão parte desta nova população é feita pelo operador genético reinserção, representado no algoritmo pelo método homônimo. A criação de novos indivíduos é feita através dos operadores genéticos seleção de pais, *crossover* e mutação. Assim, o método *geraPopulacaoFilhos()* abrange vários métodos responsáveis pela função de todos esses operadores genéticos, que no final, produz uma população inteira de cromossomos filhos.

O método *reinserção()* é responsável por escolher os cromossomos (entre os pais e filhos) que farão parte da nova geração populacional. Ambos AGs (StandardAG e o TransgenicAG) implementados neste estudo, utilizam o método de reinserção melhores pais e filhos. Portanto, a nova população de cromossomos é composta pelos cromossomos com melhor valor de aptidão entre os cromossomos pais e cromossomos filhos.

O método *executar()* também define a semente (*seed*) utilizada pelos cálculos aleatórios do AG. Várias operações de um Algoritmo Genético são aleatórias, utilizam valores sorteados. Para a realização dos experimentos, é necessário utilizar o mesmo valor de semente para que não haja alteração dos valores sorteados e permita uma análise precisa dos resultados. O método *defineSeed()* define uma semente fixa para o AG. Este método só é chamado durante os experimentos, antes da chamada do método *executar()*. Se esse método não for invocado, o AG é executado sem um valor fixo de semente definido para todas operações randômicas.

4.2 TransgenicAG

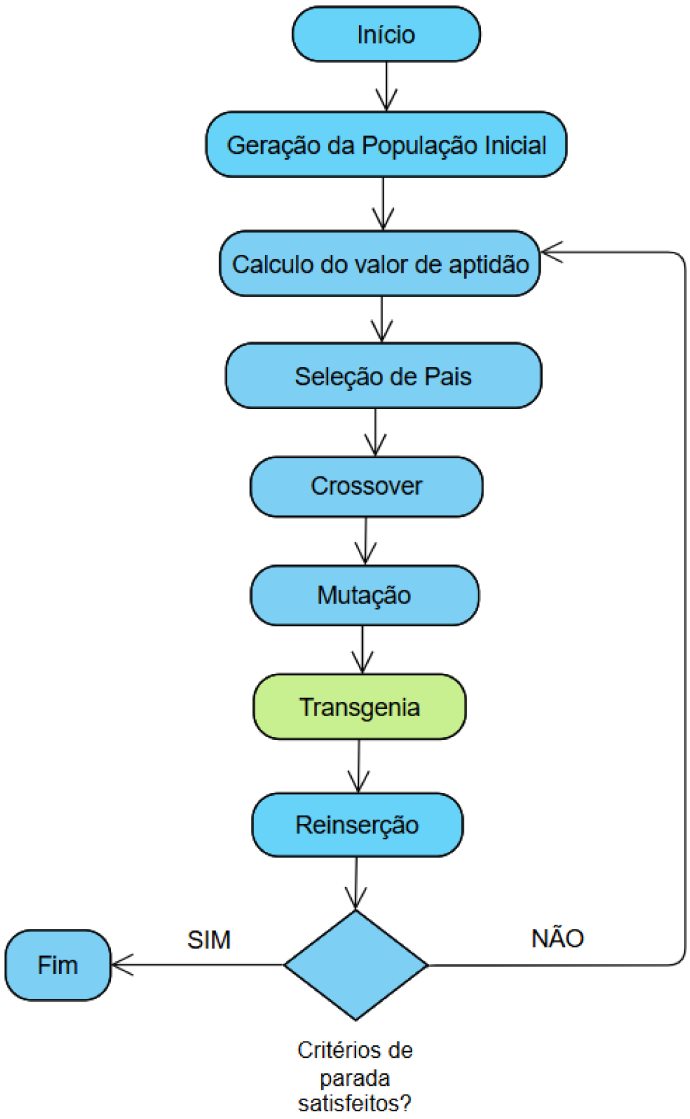
4.2.1 Conceitos

O Algoritmo Genético Transgênico é uma variação do Algoritmo Genético convencional. O Algoritmo Genético Transgênico segue o mesmo ciclo (os mesmos operadores genéticos) de um Algoritmo Genético convencional. No entanto, ele inclui um novo operador genético após a mutação e antes da reinserção, o operador *transgenic*. A Figura 14 retrata o fluxo de processos do Algoritmo Genético Transgênico.

O operador transgênico foi proposto por Amaral e Hruschka (2011). Assim como outros operadores genéticos, o operador transgênico é baseado em princípios biogenéticos, especificamente o conceito de transgênese. Por meio de processos de engenharia genética, o material genético de organismos pode ser manipulado. Genes de um organismo podem ser introduzidos diretamente no genoma de outro organismo. O gene introduzido pode se

originar da mesma espécie ou de uma espécie completamente diferente. Esse processo é chamado de transgênese ou transgenia. Essa técnica introduz novos traços ou características em indivíduos que naturalmente não têm essas características. A transgenia é utilizada para produzir organismos geneticamente modificados, que podem potencialmente ter características melhores em comparação a outros organismos da mesma espécie.

Figura 14 – Ciclo de um Algoritmo Genético Transgênico



Fonte: elaborado pelo autor

Os indivíduos alterados por técnicas de DNA recombinante são chamados organismos geneticamente modificados (OGMs). Todo organismo pode ter modificações induzidas em seu material genético, porém os OGMs são frequentemente encontrados na agricultura. Os benefícios dos transgênicos permitem aumentar a resistência de culturas, além de outros aprimoramentos, melhorando amplamente a cadeia de produção de alimentos (PIMENTEL, 1989).

Observando o potencial da transgênese, Amaral e Hruschka (2011) projetaram um novo operador genético para Algoritmos Genéticos. O objetivo do operador transgênico é encontrar a solução ótima mais rapidamente, identificando genes relevantes dentro da população e replicando os valores desses genes em indivíduos com baixo potencial. Em Algoritmos Genéticos, indivíduos com baixo potencial são cromossomos com baixos valores de aptidão. Em contrapartida, os genes de indivíduos com altos valores de aptidão são possíveis genes relevantes. Após identificar indivíduos com baixo potencial e os indivíduos com genes relevantes dentro da população, o operador transgênico escolhe aleatoriamente um ou mais genes relevantes que serão introduzidos diretamente nos indivíduos com baixo potencial. Baseado nessas premissas, o **TransgenicAG** foi implementado com a concepção de otimizar fórmulas matemáticas utilizando o operador *transgenic*.

4.2.2 Algoritmo Genético Transgênico

Como pode ser observado na Figura 14, o Algoritmo Genético Transgênico possui o mesmo fluxo de funcionamento do AG convencional, mas com a adição de um novo operador genético, a **transgenia**, baseado no operador *transgenic* de Amaral e Hruschka (2011). O Algoritmo 2 retrata o fluxo geral de operações do TransgenicAG. O trecho de Código 3 apresenta o método *executar()* e o método *definirSeed()* do Algoritmo Genético Transgênico.

Algoritmo 2 – Fluxo geral do Algoritmo Genético Transgênico

```
1: procedure TransgenicAG                          ▷ fluxo geral do TransgenicAG

2:     CriarPopulacaoInicial()
3:     continuidade ← verificaContinuidade()

4:     while continuidade = TRUE do

5:         SelecionarPaisCrossover()
6:         CrossOver()
7:         m ← verificaProbabilidadeMutacao()

8:         if m = TRUE then
9:             Mutacao()
10:        end if

11:        Transgenia()
12:        Reinsercao()
13:        continuidade ← verificaContinuidade()

14:    end while
15: end procedure
```

Código 3 – TransgenicAG Método *executar*() (Classe Transgenic.java)

```
//O metodo geraPopulacaoFilhos() serve para gerar cromossomos filhos
    derivados dos cromossmos da geracao atual de individuos
//Os operadores geneticos: selecao pais, crossover e mutacao sao
    implementados por metodos invocados dentro do metodo
    geraPopulacaoFilhos();

public int executar(MathFunction funcaoMatematica) {
    this.funcaoMatematica=funcaoMatematica;

    if(seed==null)
        r = new Random();
    else
        r = new Random(this.seed);

    inicializaPopulacao();
    verificaContinuidadeAG(this.populacao);

    while(this.continuidade){
        geraPopulacaoFilhos();
        transgenia();
        reinsercao();
        verificaContinuidadeAG(this.populacao);
    }
    return this.indexPopulacao;
}

public void definirSeed(long seed){
    this.seed=seed;
}
```

O método *executar()* do TransgenicAG segue o fluxo de operações representadas no algoritmo 2. O método *geraPopulacaoFilhos()* realiza as operações de seleção de pais, *crossover* e mutação, de forma similar ao StandardAG. Aliás, todo o método *executar()* do TransgenicAG é bem similar ao StandardAG, porém com uma diferença, a inclusão da **transgenia** (abstração do operador *transgenic* de Amaral e Hruschka).

A transgenia é aplicada após a mutação e antes da reinserção. A mutação é um operador genético condicional, desta forma, a mutação só é aplicada em cromossomos sorteados, escolhidos aleatoriamente de acordo com a taxa de mutação definida na inicialização. Assim, a transgenia pode ocorrer após o *crossover*, caso o cromossomo não sofra mutação. A aplicação do operador genético *crossover* é constante, sempre executado após a seleção de pais.

4.2.3 Operador *transgenic*

O operador *transgenic* é implementado pelo método *transgenia()* no TransgenicAG. O Algoritmo 3 demonstra passo a passo o funcionamento do operador *transgenic*. O operador *transgenic* utiliza dois novos parâmetros definidos na inicialização: a taxa de transgenia e o número de doadores. A taxa de transgenia define o percentual de piores indivíduos que irão sofrer transgenia dentro da população. O número de doadores define a quantidade dos n melhores cromossomos que são selecionados como potenciais doadores.

Inicialmente, os pais e filhos são agrupados em uma única lista. É criada uma lista de *doadores*, onde em cada operação de transgenia, será sorteado um doador desta lista. Para cada indivíduo ruim (os n piores indivíduos são selecionados, um por um, baseado no seu valor de aptidão), é sorteado um doador e também um (ou mais) gene(s) do doador. O valor deste gene sorteado é aplicado diretamente sobre o valor do gene do cromossomo receptor, sobrescrevendo o valor antigo. Para fins de explicação do funcionamento do operador *transgenic*, o algoritmo 3 retratou o processo de transgenia para um único gene, porém, este processo pode ser aplicado a dois ou mais genes. Após encerrado o processo de transgenia, o Algoritmo Genético Transgênico continua sua cadeia de processos, executando o operador genético *reinserção*. O Código 4 apresenta a implementação do processo de transgenia no TransgenicAG.

Algoritmo 3 – Operador *transgenic*

```
1:  procedure TRANSGENIA                          ▷ funcionamento da transgenia()

2:      numIndividuosRuins ← ((taxaTransgenia × qtdPopulacional) ÷ 100)
3:      doadores ← obtemListaDoadores(qtdDoadores)

4:      populacao ← AgrupaPopulacaoPaisComFilhos()
5:      i ← 0

6:      while i ≤ numIndividuosRuins do

7:          receptor ← populacao[qtdPopulacional − (i + 1)]
8:          doadorSorteado ← sorteiaDoador()
9:          geneSorteado ← sorteiaGene(receptor)
10:         receptor[geneSorteado] ← doadorSorteado[geneSorteado]
11:         recalculaValorFitness(receptor)

12:         i ← (i + 1)

13:     end while

14: end procedure
```

Código 4 – TransgenicAG Método *transgenia*() (Classe Transgenic.java)

```
public void transgenia(){

    this.populacao.addAll(this.populacaoFilhos);
    this.populacao.sort(null);

    int numero_individus_ruins = (int) Math.floor((this.taxa_transgenia*
   this.qtdPopulucao)/100);

    Cromossomo[] doadores   = new Cromossomo[this.numero_doadores];

    for(int i=0;i<this.numero_doadores;i++){
        doadores[i]=this.populacao.get(i);
    }

    if(this.printTransgenia)
        System.out.println("Receptores Geracao: "+indexPopulacao);

    for(int i=0;i<numero_individus_ruins;i++){

        Cromossomo receptor=this.populacao.get((this.qtdPopulucao-(i+1))
   );
        Cromossomo doador_sorteado= doadores[this.r.nextInt(doadores.
   length)];
        int indice_gene_sorteado = this.r.nextInt(doador_sorteado.
   getGenes().length);
        receptor.getGenes()[indice_gene_sorteado]= doador_sorteado.
   getGenes()[indice_gene_sorteado];
        receptor.setFx(funcaoMatematica.calcular(receptor.getGenes()));
        receptor.setAvaliacao(funcaoMatematica.fitnessFunction(receptor)
   );
        this.populacao.set(this.qtdPopulucao-(i+1), receptor);
                    countTransgenia++;

   }
}
```

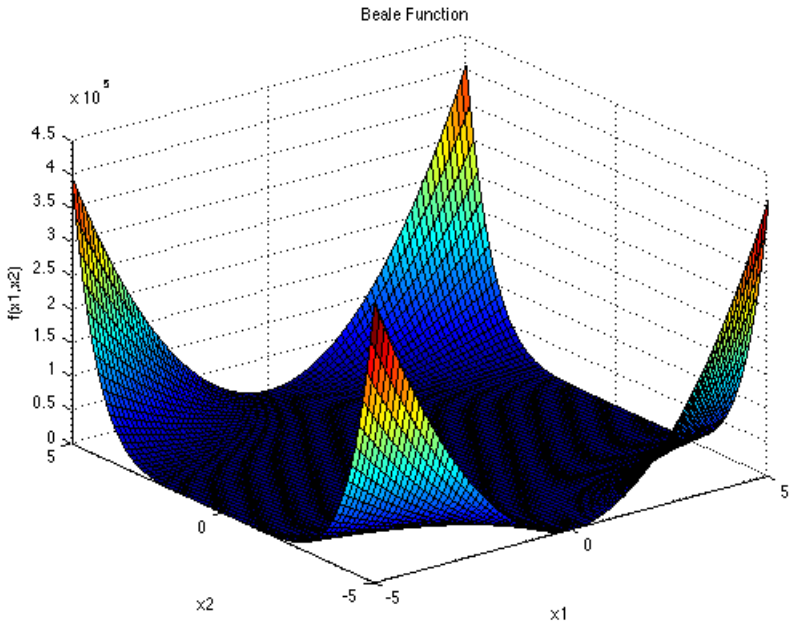
O objetivo principal do operador *transgenic* é fazer com que o Algoritmo Genético Transgênico encontre o valor ótimo com menos gerações populacionais do que um Algoritmo Genético convencional demoraria para encontrar. No caso das fórmulas matemáticas, o valor ótimo corresponde ao mínimo ou máximo global (dependendo da fórmula matemática). Dependendo do problema que são utilizados para otimizar, Algoritmos Genéticos podem consumir bastante recursos computacionais. Desse modo, a redução de gerações pode oferecer ganhos computacionais consideráveis em processamento e memória.

4.3 Equações matemáticas

Os algoritmos StandardAG e TransgenicAG foram desenvolvidos com o propósito de otimizar equações matemáticas. Ambos os algoritmos genéticos podem ser utilizados para tentar otimizar qualquer fórmula matemática. Porém, para fins deste experimento, três funções matemáticas diferentes foram escolhidas para serem utilizadas nos experimentos por diferentes razões. As seguintes funções matemáticas foram escolhidas para os experimentos: *Beale*, *McCormick* e *Michalewicz*.

4.3.1 Função Beale

Figura 15 – Função Beale



Fonte: Bingham e Surjanovic (2013)

A Figura 15 ilustra a função de *Beale*, uma função multimodal com picos acentuados nos cantos. Em geral, é uma função simples para otimizar. Ela possui um mínimo global. A função de *Beale* é descrita abaixo:

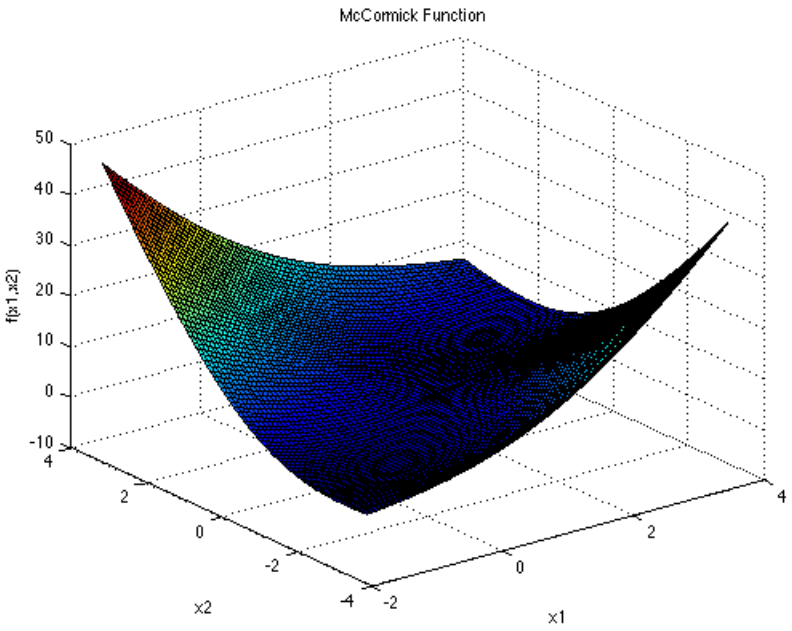
$$f(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$$

Mínimo Global: $f(x^*) = 0$ em $x^* = (3, 0.5)$
Os dados de entrada da função estão dentro do intervalo [-4.5, 4.5]

Os dados pertinentes à função *Beale* foram extraídos do site da biblioteca virtual de experimentos de simulação da Universidade de Simon Fraser: *https://www.sfu.ca/ ssurjano/beale.html*.

4.3.2 Função McCormick

Figura 16 – Função McCormick



Fonte: Bingham e Surjanovic (2013)

A Figura 16 representa a função *McCormick*, uma função com características semelhantes à função *Beale*, no entanto, diferentemente da função *Beale*, ela foi escolhida porque seus valores de entrada e o seu mínimo global possuem cinco casas decimais. A dificuldade de otimização dessa função será a precisão, pois números decimais com cinco casas de precisão aumentam exponencialmente o espaço de busca. Esta função será útil para avaliar a precisão dos AGs ao lidar com várias casas decimais. A função *McCormick* é descrita abaixo:

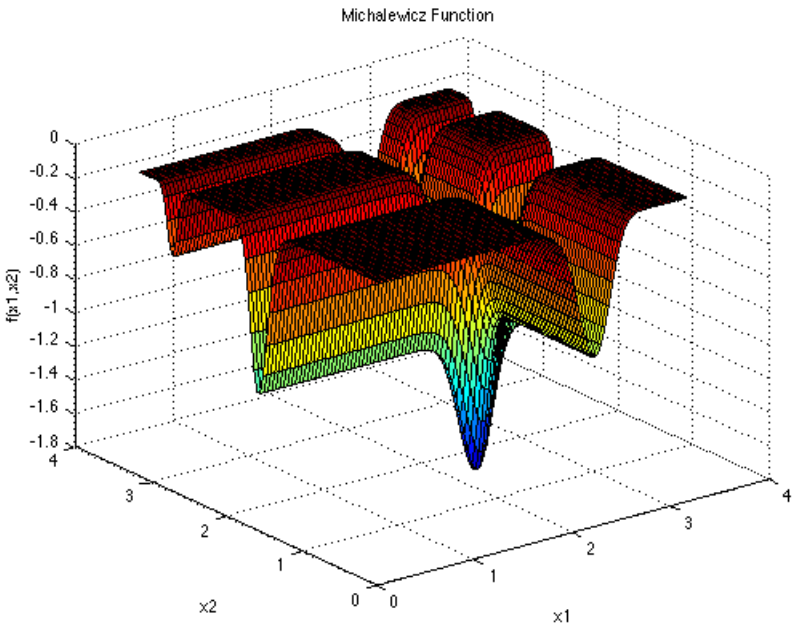
$$f(x) = \sin(x_1 + x_2) + (x_1 - x_2)^2 - 1.5x_1 + 2.5x_2 + 1$$

Mínimo Global: $f(x^*) = -1.9133$ em $x^* = (-0.54719, -1.54719)$
Os dados de entrada da função estão dentro do intervalo: $x_1 \in [-1.5, 4]$, $x_2 \in [-3, 4]$

Os dados pertinentes à função *McCormick* foram extraídos do site da biblioteca virtual de experimentos de simulação da Universidade de Simon Fraser: *https://www.sfu.ca/ ssurjano/mccorm.html*.

4.3.3 Função Michalewicz

Figura 17 – Função Michalewicz



Fonte: Bingham e Surjanovic (2013)

A Função *Michalewicz* é uma função matemática utilizada majoritariamente para testes de otimização de algoritmos de busca e heurísticas, sendo altamente multimodal, possuindo múltiplos mínimos locais. O valor do mínimo global varia de acordo com o valor do parâmetro d. O parâmetro d indica o número de variáveis utilizadas na função. Outro parâmetro presente na função é o parâmetro m, que indica a complexidade e dificuldade de otimização da função. Quanto maior o valor de m, maior a dificuldade de convergência da função *Michalewicz*. Caso não informado, o valor padrão de m é $m = 10$. A Figura 17 retrata a função Michalewicz.

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

Mínimo Global em d = 2: $f(x^*) = -1.8013$ em $x^* = (2.20, 1.57)$

Geralmente, a função é avaliada $xi \in [0, \pi]$, para todo i= 1, . . . , d.

Os dados pertinentes à função *Michalewicz* foram extraídos do site da biblioteca virtual de experimentos de simulação da Universidade de Simon Fraser: *https://www.sfu.ca/ ssur-*

jano/michal.html.

Os experimentos realizados com a função *Michalewicz* nesse estudo utilizaram os seguintes valores para os parâmetros m e d:

- $m = 10$
- $d = 2$

Durante os experimentos foi observado um aumento crítico na complexidade de otimização da função *Michalewicz* utilizando mais do que dois genes ($d > 2$) e com um valor acima do valor padrão para o parâmetro m ($m > 10$), exigindo populações com número de cromossomos bem discrepantes em relação ao tamanho das populações definidas para os experimentos com as funções *Beale* e *McCormick*. Nos testes realizados com valores acima de 2 e 10 para d e m respectivamente, não houve nenhum único resultado com convergência, tanto com o StandardAG quanto com o TransgenicAG.

Ao utilizar o valor padrão do parâmetro m ($m = 10$) e dois genes (d= 2), os experimentos da função *Michalewicz* obtiveram resultados com convergência utilizando populações com o mesmo número de cromossomos definidos nos experimentos das funções *Beale* e *McCormick*.

Capítulo 5

Experimentos

Este capítulo apresenta uma visão geral dos experimentos realizados nesta pesquisa. A finalidade principal dos experimentos é permitir idealizar uma comparação entre o Algoritmo Genético convencional e o Algoritmo Genético Transgênico na otimização de fórmulas matemáticas.

5.1 Visão geral

O Algoritmo Genético Transgênico foi projetado como uma nova abordagem de algoritmo evolucionário, com base nos avanços do mundo real na tecnologia de DNA recombinante. Espera-se que o Algoritmo Genético Transgênico possa ser tão eficaz quanto ou superior a um Algoritmo Genético convencional na busca de soluções ótimas em diferentes contextos de situações-problema. Algoritmos evolucionários surgiram com o desafio de otimizar problemas de diferentes graus de complexidade. Problemas com múltiplos picos locais, espaços de busca complexos e ótimos globais difíceis de identificar tendem a ser problemas de alta complexidade.

Para a realização deste estudo comparativo, duas versões de Algoritmos Genéticos foram desenvolvidas. O StandardAG foi projetado como uma implementação do Algoritmo Genético convencional. O TransgenicAG foi idealizado como uma variação ao AG convencional, utilizando o operador *transgenic* de Amaral e Hruschka (2011). Ambos foram implementados para que seja possível elaborar um estudo comparativo entre o AG convencional e o AG transgênico. Assim, este estudo realiza uma comparação de desempenho, qualidade e acurácia entre um Algoritmo Genético convencional e o Algoritmo Genético transgênico. Em outras palavras, este experimento propõe mensurar o quanto o operador *transgenic* pode agregar eficiência e eficácia na busca de ótimos globais de equações matemáticas. Desse modo, espera-se ser possível dimensionar o quanto a transgenia pode agregar valor à convergência de Algoritmos Genéticos na otimização de equações matemáticas.

5.2 Método para a Avaliação

5.2.1 Ambiente de testes

Tanto o Algoritmo Genético convencional (StandardAG) quanto o Algoritmo Genético Transgênico (TransgenicAG) foram codificados manualmente e implementados para este experimento. Ambos os algoritmos foram codificados na Linguagem de Programação Java, na versão 21 (a versão mais atual com Long-Term Support (LTS) até o momento da realização deste trabalho).

Todos os experimentos realizados foram executados no mesmo computador. O computador segue o padrão IBM PC do tipo Desktop com processador Intel(R) Core(TM) i7-10700 CPU @ 2.90 GHz de 8 cores e 16 threads, memória RAM de 16 GB Dual-Channel DDR4 @ 1463MHz, com duas unidades de armazenamento. Um disco SSD NVMe de 256GB com taxas de leitura e escrita sequenciais de 1410 MBps e 850 MBps, com o sistema operacional Windows 10 Pro 64-bit instalado e um disco SSD WDC WD10SPSX-75A6WT0 de 1TB com taxas de leitura e escrita sequenciais de 200 MBps e 170 MBps, utilizado para backup de dados.

É importante destacar que os experimentos executados nessa pesquisa podem apresentar resultados diferentes dos resultados aqui apresentados (mais especificamente, os resultados de métricas que envolvem tempo de execução), caso sejam executados em outro computador com diferente poder computacional.

5.2.2 Panorama - condições e restrições do experimento

Para realizar este experimento, é necessário considerar as condições iniciais que permitam avaliar a qualidade, precisão e desempenho do Algoritmo Genético convencional (StandardAG) e o Algoritmo Genético transgênico (TransgenicAG). Além disso, foi necessário definir as equações matemáticas usadas no experimento.

Para cada uma das equações matemáticas, uma bateria de testes foi realizada. Cada unidade de teste consistiu em executar o StandardAG e o TransgenicAG cem vezes com os mesmos parâmetros de entrada. Três unidades de teste diferentes foram realizadas para cada equação matemática usando populações de 100, 200 e 1000 cromossomos respectivamente com taxas de mutação diferentes. O experimento foi executado com duas taxas de mutação: 3% e 5% para cada AG, equação matemática e quantidade de cromossomos por população.

Além disso, é importante destacar a conjuntura das equações matemáticas durante os testes. Os testes com a função *Beale* consideraram meta de 100% na otimização, ou seja, a sua otimização ocorre quando o AG encontra o valor exato do mínimo global da função. Já as funções *Michalewicz* e *McCormick* consideraram a meta de 99% de valor de aptidão na otimização. A razão é que em ambas funções, os mínimos globais possuem 4 e

5 casas decimais respectivamente. Durante os testes, foi identificado um baixíssimo índice de sucesso na convergência destas funções com meta de 100% (valor preciso do mínimo global) devido ao vasto espaço de busca causado pelo número de casas decimais. Com meta de 99%, somente valores bem próximos são considerados valores ótimos. A meta de 99% gerou resultados com valores mais satisfatórios que permitissem uma analise sólida e mais apurada.

Outra observação importante identificada durante os experimentos, foi a dificuldade de otimização da função *Michalewicz* utilizando mais do que dois genes. Não houve resultados com convergência utilizando mais de dois genes e as populações de 100, 200 e 1000 cromossomos. Para atingir resultados com convergência utilizando três ou mais genes na função *Michalewicz*, seria necessário utilizar populações com número de cromossomos bem maiores do que o tamanho das populações estipuladas para os experimentos. Utilizar populações com tamanho bem maiores do que 100,200 e 1000 cromossomos, geraria resultados destoantes nas outras duas funções: *Beale* e *McCormick* (visto que praticamente todas as execuções iriam convergir), o que impossibilitaria traçar uma avaliação homogênea e apropriada para as três funções utilizando os mesmos parâmetros e cenários de otimização.

5.2.3 Parâmetros de inicialização

Os Algoritmos Genéticos utilizam vários parâmetros durante a inicialização que impactam no seu comportamento e performance durante a execução. A primeira etapa a ser planejada para a realização dos experimentos, foi a definição dos valores dos parâmetros de inicialização. Nesta seção, cito os parâmetros de inicialização e seus respectivos valores, definidos e utilizados nos experimentos com ambos algoritmos (StandardAG e TransgenicAG). Alguns parâmetros citados aqui são exclusivos do Algoritmo Genético Transgênico (TransgenicAG). Esses casos serão explicitamente mencionados como configuração específica do Algoritmo Genético Transgênico. Os demais parâmetros são utilizados para ambos os algoritmos.

1. Número máximo de gerações = 100

O **número máximo de gerações** determina o número máximo de populações que o algoritmo criará se não encontrar a solução ótima antes. Após atingir esse número, o algoritmo é encerrado. Para fins de análise, quando uma execução de algoritmo falha em encontrar a solução ótima, o valor de 100 gerações é computado.

2. Valor de incremento ou decremento de mutação: 0,1

Durante a execução do Algoritmo Genético, a operação de mutação será realizada de acordo com a taxa de mutação. Nos casos em que ocorre mutação, **o valor de incremento ou decremento** determina o valor que será incrementado ou decrementado para o gene. A escolha entre incrementar ou decrementar o valor do gene também é aleatória.

3. Taxa de mutação = 3% ou 5%

A **taxa de mutação** define a probabilidade de mutação ocorrer após o *crossover*. A mutação é uma operação de ajuste fino; altas taxas podem resultar em um distanciamento da solução ótima no espaço de busca quando o algoritmo começa a se aproximar dela. Os experimentos utilizam taxas de 3% e 5% de probabilidade de mutação.

4. Taxa Transgênica = 20

A taxa transgênica ou taxa de transgenia é um parâmetro exclusivo do Algoritmo Genético Transgênico (TransgenicAG). A **taxa transgênica** define o percentual de cromossomos (relacionado à quantidade populacional) com baixa avaliação que sofrerão transgênese, tendo aleatoriamente um ou mais genes alterados à força por um valor de gene relevante identificado. Para os experimentos, foi definida a taxa de 20 por cento. Assim, para as quantidades de 100, 200 e 1000 cromossomos, os 20, 40 e 200 piores cromossomos sofrerão transgênese, respectivamente.

5. Número de doadores = 5

Outro parâmetro exclusivo do Algoritmo Genético Transgênico (TransgenicAG). O parâmetro **número de doadores** define o número de cromossomos em cada geração que serão usados para fornecer genes relevantes para os piores cromossomos definidos pela taxa de transgenia. Neste experimento, apenas os cinco melhores cromossomos serão doadores de genes relevantes.

Outro aspecto importante a ser mencionado é que Algoritmos Genéticos são algoritmos não determinísticos. Em outras palavras, em cada execução, mesmo com os mesmos parâmetros, os resultados são diferentes. Isso acontece porque há várias operações aleatórias no Algoritmo Genético. Para realizar o experimento, foi necessário definir a mesma semente para todas as operações aleatórias. A mesma semente permite que tanto o Algoritmo Genético convencional quanto o Algoritmo Genético Transgênico lidem com os mesmos dados durante a execução. Isso é muito importante para avaliar os resultados, pois ambos precisam trabalhar com os mesmos dados. Em cada execução de cada experimento, o valor da semente é incrementado em 1, sendo o valor inicial (da primeira execução)

também definido como 1. Portanto, o valor da semente variará de 1 a 100 durante cada experimento realizado.

5.2.4 Métricas de Avaliação

Antes da realização de qualquer experimento, fez-se necessário determinar bons critérios e métricas que permitissem avaliar a qualidade de ambos algoritmos na otimização de cada função matemática. Durante a execução do experimento, os dados que impactam os critérios escolhidos (que serão detalhados a seguir) foram escritos em um arquivo de texto durante cada uma das cem execuções de cada um dos experimentos para análise. No início, nos primeiros testes para definir os critérios e condições do experimento, apenas o número de gerações que cada execução levou para encontrar a solução ótima e a respectiva semente utilizada na execução eram escritos. Posteriormente, foi adicionado o tempo de cada uma das execuções. Assim, as métricas de avaliação determinadas são baseadas no número de gerações e no tempo total de cada execução de cada experimento. Se a execução falhar em encontrar a solução ótima, o valor máximo de gerações (no caso 100) é registrado. Para um melhor entendimento, explicar as métricas de avaliação escolhidas e o motivo de escolha de cada uma, uma fração dos experimentos será demonstrada abaixo.

A Tabela 2 traz uma amostra das dez primeiras linhas dos resultados da função *Beale* com população de 1000 cromossomos. A coluna *semente* exibe a semente (seed) utilizada nas execuções do StandardAG e TransgenicAG com as respectivas taxas de mutação especificadas pelo cabeçalho das colunas ao lado. A coluna *Standard mut 3%* se refere à quantidade de gerações que o StandardAG com taxa de mutação de 3% gastou para convergir. Da mesma forma, a coluna *Transgenic mut 3%* se refere à quantidade de gerações que o TransgenicAG com taxa de mutação de 3% gastou para convergir. Tanto o AG transgênico quanto o AG convencional foram executados também com taxa de mutação de 5%, onde as respectivas colunas com o nome do AG e taxa de mutação de 5% referenciam. As colunas *tempo* exibem o tempo gasto na execução do respectivo AG (indicado pela primeira coluna à esquerda de cada coluna tempo) com a respectiva semente utilizada.

A primeira métrica capaz de avaliar e distinguir o AG convencional e o AG transgênico, é o número total de gerações populacionais que o AG leva para convergir. Cada tabela gerada pelos experimentos possui cem linhas, contendo os resultados de cem execuções do Algoritmo Genético convencional e do Algoritmo Genético transgênico, utilizando mutações de 3% e 5%. Quando o AG converge, o número de gerações necessárias para convergir durante essa execução é exibido na respectiva célula. No exemplo da Tabela 2, a semente 8, é o único caso em que somente o StandardAG não convergiu (na amostra, o TransgenicAG convergiu em todas as dez execuções). Quando o AG não consegue convergir em uma execução, o número máximo de gerações é exibido na respectiva célula. Como pode ser visto, o StandardAG em ambas taxas de mutações gastou o número máximo de gerações com a semente 8, indicando que ele atingiu o número máximo de gerações

Tabela 2 – Amostra de dados coletados do teste de função de Beale

Semente	Standard mut 3%	Tempo	Standard mut 5%	Tempo	Transgenic mut 3%	Tempo	Transgenic mut 5%	Tempo
1	3	148	3	154	3	148	3	111
2	4	55	2	59	3	111	2	86
3	2	11	2	20	2	24	2	11
4	2	10	2	26	2	10	2	12
5	3	20	3	31	5	35	5	34
6	2	11	2	15	2	12	2	12
7	8	143	8	126	2	12	2	11
8	100	1638	100	1591	4	24	2	11
9	5	30	5	37	3	16	2	9
10	2	8	2	10	2	9	2	10

definido na parametrização dos AGs. Em todas as outras linhas, em todos os cenários, os algoritmos convergiram gastando um número diferente de gerações populacionais durante a execução. A listagem completa dos resultados individuais, com as informações das cem execuções de cada experimento, está no apêndice A, capítulo 7.

Inicialmente, como métrica para avaliar o número de gerações, foi planejado o uso da média de gerações convergentes. Além disso, o desvio padrão e os intervalos de confiança também foram planejados como medidas para mensurar o potencial de cada AG em cada diferente cenário (função matemática testada, diferentes taxas de mutação e experimentos com diferentes números de cromossomos por população). No entanto, foi observado que essas medidas mascaram a real capacidade de convergência dos AGs, gerando resultados que não permitiriam mensurar a qualidade real de cada algoritmo em cada contexto de experimento.

Desse modo, as melhores medidas encontradas para analisar a capacidade de convergência dos AGs através dos resultados dos experimentos são: o número total de execuções convergentes e o número médio de gerações que o AG avaliado precisa para convergir.

O número total de execuções convergentes é uma boa métrica de avaliação, porque quanto maior o número de execuções convergentes, demonstra o quanto o determinado AG foi mais eficaz do que o outro naquele cenário.

No entanto, mesmo que um AG tenha um bom índice de convergência (converja mais vezes que o outro AG) ele pode não ser eficiente, pois ele pode necessitar de mais gerações populacionais para convergir do que o outro. Enquanto que o outro AG pode convergir mais rápido, gastando em média menos gerações para encontrar a solução ótima. Para mensurar essa condição, a média de gerações que um AG leva para convergir também será considerada na análise. Assim, a capacidade de convergência do TransgenicAG e do StandardAG é avaliada pelo número total de execuções convergentes e a média de gerações que cada AG leva para convergir.

Outro fator que pode proporcionar boas métricas de avaliação é o tempo de execução.

Em um experimento qualquer entre os experimentos realizados, para cada uma das cem execuções, o tempo total que o AG foi executado é registrado em milissegundos. O registro do tempo de execução do AG permite a análise por outras métricas de eficiência. Assim, o AG convencional (StandardAG) e AG transgênico (TransgenicAG) também são avaliados pelo tempo total de execução de cada experimento, o tempo médio das execuções que convergiram e o tempo médio das execuções que não convergiram.

5.3 Experimentos

O planejamento dos experimentos ocorreu após a definição das melhores métricas que permitissem avaliar com consistência as diferentes qualidades e atributos de cada um dos AGs em diferentes contextos de otimização. Dessa maneira, as etapas e condições dos experimentos foram definidas.

Os experimentos foram realizados da seguinte forma: Para cada uma das três funções matemáticas escolhidas (*Beale*, *McCormick* e *Michalewicz*), os dois AGs desenvolvidos (StandardAG e TransgenicAG) foram executados cem vezes utilizando uma semente (seed) diferente (que varia de 1 a 100). Cada experimento utiliza uma quantidade populacional de cromossomos diferente e uma taxa de mutação diferente. Foram realizados experimentos com três diferentes populações: populações de 100, 200 e 1000 cromossomos. Foram realizados experimentos com duas taxas diferentes de mutação: 3% e 5%. Cada experimento registra as seguintes informações de cada execução em uma linha de um arquivo de texto: semente, a quantidade de gerações que o AG levou para convergir; e o tempo total da execução. Foi definido o parâmetro máximo de gerações de 100 gerações, portanto, as execuções que não convergiram são identificadas por registrarem 100 gerações. Os demais parâmetros utilizados nos AGs possuem valores idênticos para todos os experimentos em todos os contextos. Os valores padrão atribuídos aos parâmetros são apresentados na seção 5.2.3.

Cada experimento gera um arquivo com cem registros (uma linha para cada execução com as informações de semente, quantidade de gerações que o AG levou para convergir e o tempo de execução). O arquivo gerado por cada experimento expressa os resultados individuais de cada uma das cem execuções. Os resultados individuais completos de cada experimento podem ser verificados no Apêndice A, no capítulo 7 em forma de tabelas. Os resultados individuais são divididos por função matemática e tamanho populacional utilizados no experimento. Para diminuir a quantidade de tabelas, os resultados com as taxas de mutação de 3% e 5% de ambos AGs (StandardAG e TransgenicAG) foram mesclados em uma única tabela. Assim, cada resultado individual apresenta os resultados do StandardAG e do TransgenicAG, nas taxas de mutação de 3% e 5% por função matemática e tamanho populacional.

Após a obtenção dos resultados individuais de cada experimento, foram feitos cálcu-

los para adaptar os dados coletados às métricas de avaliação definidas na seção 5.2.4. Para facilitar a visualização, análise e proporcionar clareza, os resultados da análise por métricas de avaliação foram condensados em tabelas, sendo uma tabela por função matemática (*Beale*, *McCormick* e *Michalewicz*). A seguir, são apresentados os resultados dos experimentos:

Tabela 3 – Resultados dos experimentos da função Beale

Métricas	População	StandardAG mutação 3%	StandardAG mutação 5%	TransgenicAG mutação 3%	TransgenicAG mutação 5%
Quantidade de execuções convergentes	100 Cromossomos	14	14	34	35
	200 Cromossomos	35	36	62	67
	1000 Cromossomos	93	92	95	98
Número médio de gerações necessárias para encontrar a solução ótima	100 Cromossomos	4,42	4,42	8,70	7,6
	200 Cromossomos	5,02	5,19	6,5	6,28
	1000 Cromossomos	3,90	3,90	3,64	3,19
Tempo total de execução (milissegundos)	100 Cromossomos	3396	3295	2498	2417
	200 Cromossomos	7100	6931	4645	3826
	1000 Cromossomos	15408	17219	12066	5642
Média de tempo das execuções que convergiram (milissegundos)	100 Cromossomos	5,21	5,42	2,88	1,88
	200 Cromossomos	4,2	4,27	3,95	3,39
	1000 Cromossomos	29,48	32,95	25,45	18,26
Média de tempo das execuções que não convergiram (milissegundos)	100 Cromossomos	38,63	37,43	36,36	36,17
	200 Cromossomos	106,96	105,89	115,79	109,06
	1000 Cromossomos	1809,42	1773,37	1929,6	1926

Tabela 4 – Resultados dos experimentos da função McCormick

Métricas	População	StandardAG mutação 3%	StandardAG mutação 5%	TransgenicAG mutação 3%	TransgenicAG mutação 5%
Quantidade de execuções convergentes	100 Cromossomos	2	2	9	10
	200 Cromossomos	6	6	20	18
	1000 Cromossomos	58	59	83	89
Número médio de gerações necessárias para encontrar a solução ótima	100 Cromossomos	3	3	5,78	6,5
	200 Cromossomos	3,83	3,83	5,5	5,11
	1000 Cromossomos	4,03	4,15	3,54	3,80
Tempo total de execução (milissegundos)	100 Cromossomos	6298	6211	4579	4652
	200 Cromossomos	15133	15143	11449	11467
	1000 Cromossomos	91791	86802	33793	23887
Média de tempo das execuções que convergiram (milissegundos)	100 Cromossomos	2	2,5	1,89	1,9
	200 Cromossomos	4,33	4	3,95	3,78
	1000 Cromossomos	39,39	41,03	25,30	27,22
Média de tempo das execuções que não convergiram (milissegundos)	100 Cromossomos	64,22	63,32	50,13	51,48
	200 Cromossomos	160,71	160,84	142,12	139,01
	1000 Cromossomos	2131,09	2058,07	1864,29	1951,27

Tabela 5 – Resultados dos experimentos da função Michalewicz

Métricas	População	StandardAG mutação 3%	StandardAG mutação 5%	TransgenicAG mutação 3%	TransgenicAG mutação 5%
Quantidade de execuções convergentes	100 Cromossomos	44	43	61	64
	200 Cromossomos	76	75	83	92
	1000 Cromossomos	100	100	100	100
Número médio de gerações necessárias para encontrar a solução ótima	100 Cromossomos	5,06	4,97	4,49	4,53
	200 Cromossomos	4,68	4,73	4,01	4,25
	1000 Cromossomos	3,16	3,17	2,9	2,91
Tempo total de execução (milissegundos)	100 Cromossomos	2283	2352	1299	1137
	200 Cromossomos	3289	3379	1972	1129
	1000 Cromossomos	2249	2204	1651	1693
Média de tempo das execuções que convergiram (milissegundos)	100 Cromossomos	2,40	2,39	1,41	1,19
	200 Cromossomos	2,84	2,98	2,22	2,09
	1000 Cromossomos	22,49	22,04	16,51	16,93
Média de tempo das execuções que não convergiram (milissegundos)	100 Cromossomos	38,87	39,45	31,10	29,47
	200 Cromossomos	128,04	126,2	105,18	117,12
	1000 Cromossomos	-	-	-	-

5.4 Avaliação dos Resultados

Uma observação bem clara que podemos inferir ao analisar a quantidade de execuções que convergiram em todos os resultados é que a convergência melhora ao aumentar a população. Esse resultado é esperado, pois populações maiores significam uma quantidade maior de soluções candidatas e consequentemente, uma expansão do espaço de busca. Portanto, independente de qual AG é utilizado (o AG convencional ou AG transgênico) na otimização, o aumento populacional aumenta também a probabilidade de sucesso de um Algoritmo Genético encontrar a solução ótima.

De acordo com Goldberg (1989), os Algoritmos Genéticos podem consumir muitos recursos computacionais, dependendo da complexidade do problema e do tamanho da população de soluções candidatas. Ganhos computacionais de memória, tempo de processamento e poder computacional são valiosos pois a natureza de problemas que os Algoritmos Genéticos são especializados em resolver tem complexidade variável, podendo em alguns casos ser problemas de alta complexidade. De um modo geral, o AG transgênico apresentou resultados expressivamente melhores que o AG convencional no tocante a convergência e às métricas de tempo de execução.

Uma análise individual dos resultados do StandardAG e do TransgenicAG por função matemática otimizada permite identificar conclusões mais precisas. As subseções 5.4.1, 5.4.2 e 5.4.3 apresentam análises individuais por função matemática dos resultados obtidos. É importante observar que os cálculos percentuais apresentados nas análises feitas nas subseções 5.4.1, 5.4.2 e 5.4.3 são realizados sobre o melhor resultado do TransgenicAG em relação ao melhor resultado do StandardAG e vice-versa (dependendo de qual AG obteve o melhor resultado na métrica discutida).

5.4.1 Função Beale

5.4.1.1 Comparativo entre o StandardAG x TransgenicAG

No geral, o TransgenicAG apresentou resultados melhores do que o StandardAG na maioria das métricas de avaliação. Com população de 100 cromossomos, o TransgenicAG com taxa de mutação de 5% convergiu 35 vezes, enquanto o StandardAG convergiu 14 vezes em ambas as taxas de mutação de 3% e 5%. Já os experimentos com população de 200 cromossomos, o TransgenicAG com taxa de mutação de 5% convergiu 67 vezes, enquanto que o StandardAG, no melhor cenário, convergiu 36 vezes. Percentualmente, o TransgenicAG convergiu 150% e 86,11% mais vezes do que o StandardAG nas populações de 100 e 200 cromossomos respectivamente. Na população de 1000 cromossomos, o TransgenicAG convergiu mais vezes também que o StandardAG, porém por uma vantagem pouco significativa, 98 convergências para o TransgenicAG em comparação a 93 convergências do StandardAG. A Figura 18 retrata esse cenário.

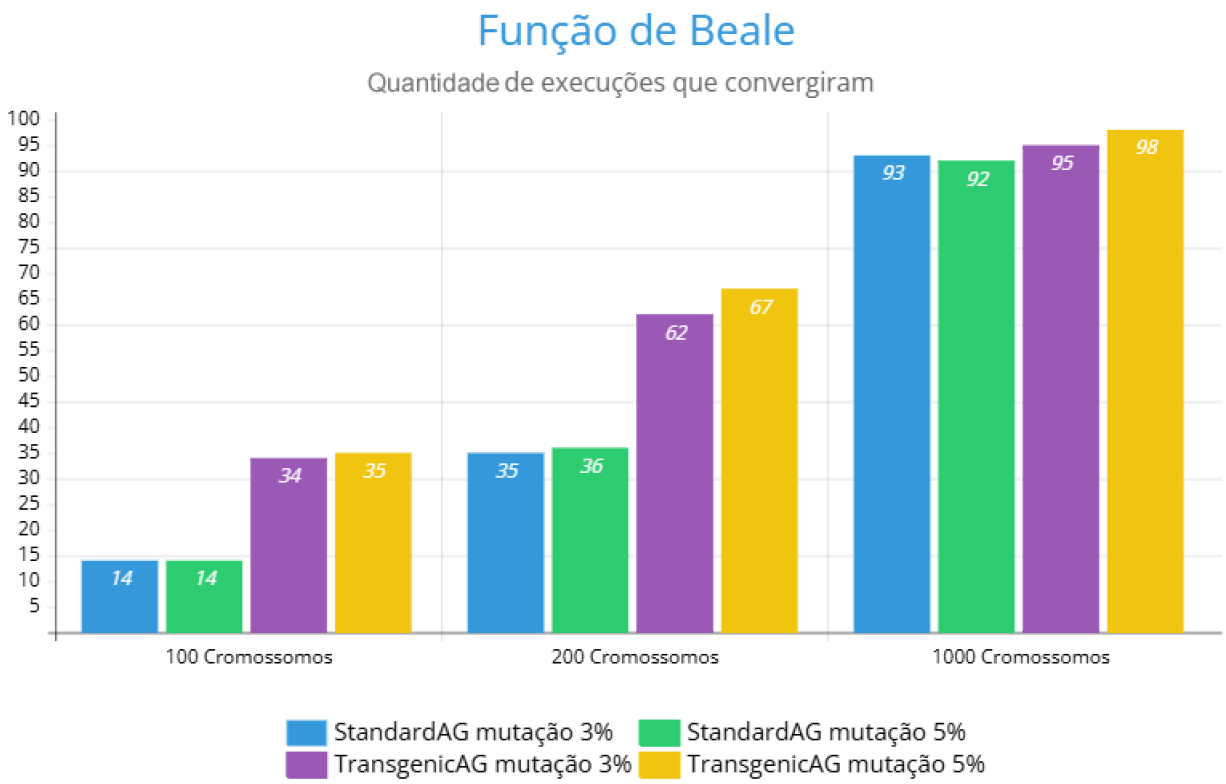


Figura 18 – Quantidade de execuções que encontram a solução ótima - Função *Beale*

Fonte: elaborado pelo autor

Ao analisar a média de gerações que os AGs levaram para convergir, o StandardAG teve o melhor desempenho em todos os cenários: 4,42, 5,02 e 3,9 gerações em média respectivamente, um ganho de 41,84%, 20,06% e 18,21% em relação ao melhor resultado do TransgenicAG.

Em relação ao tempo total de execução, o TransgenicAG teve melhor desempenho em comparação ao StandardAG em todos os cenários. O TransgenicAG foi 26,65%, 44,80% e 63,38% mais rápido que o StandardAG nas populações de 100, 200 e 1000 cromossomos. Da mesma forma, na média de tempo das execuções que convergiram, o TransgenicAG apresentou resultados superiores. O TransgenicAG foi 63,92%, 19,29%, 38,06% mais rápido do que o melhor resultado do StandardAG nas populações de 100, 200 e 1000 cromossomos. Contudo, em relação ao tempo médio das execuções com gerações que não convergiram, o StandardAG foi mais rápido nas populações de 200 e 1000 cromossomos, enquanto o TransgenicAG foi mais rápido apenas na população de 100 cromossomos.

5.4.1.2 Comparativo por taxa de mutação

Na função de *Beale*, a maioria dos resultados não apresentaram uma grande diferença entre utilizar uma taxa de mutação de 3% e 5%. A única métrica de avaliação que demonstrou um grande impacto ao utilizar mutação de 5% em relação à mutação de 3% foi

o tempo total de execução. A Figura 19 apresenta os resultados do tempo total de execução dos experimentos com a função *Beale* graficamente. Como pode ser observado pela Figura 19, a mutação de 5% reduz drasticamente o tempo de execução, principalmente com uso do Algoritmo Transgênico. O impacto é maior com o aumento populacional. Com 1000 cromossomos, o TransgenicAG com taxa de mutação de 5% consumiu 53,24% menos tempo de execução do que o TransgengicAG com taxa de mutação de 3%. Nas outras métricas de avaliação, o uso de diferentes taxas de mutação produziu pouco impacto nos resultados.

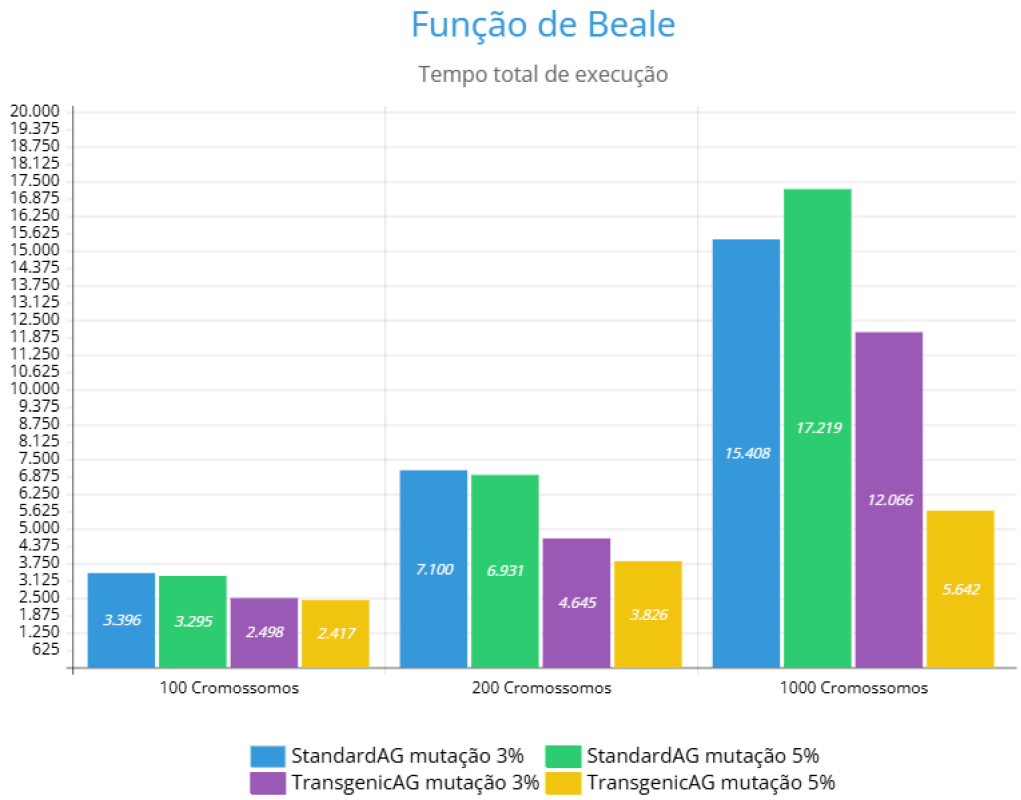


Figura 19 – Tempo total de execução - função *Beale*

Fonte: elaborado pelo autor

5.4.2 Função McCormick

5.4.2.1 Comparativo entre o StandardAG x TransgenicAG

Ao comparar os resultados do TransgenicAG com o StandardAG, o TransgenicAG apresentou resultados superiores tanto em desempenho (tempo de processamento) quanto em número de execuções que convergiram.

Nas métricas de desempenho, relacionadas ao tempo de processamento, o TransgenicAG foi superior em todos os cenários de experimentos da função *McCormick*. Analisando o tempo médio das execuções que convergiram, o TransgenicAG teve melhor desempe-

nho do que o StandardAG em todos os cenários, com uma boa vantagem percentual. Os melhores resultados do TransgenicAG foi 5,50%, 5,50% e 35,77% superiores aos melhores resultados do StandardAG nas populações de 100, 200 e 1000 cromossomos, respectivamente. Da mesma forma, ao analisar somente as execuções que não convergiram, o TransgenicAG também obteve menor tempo de processamento em média. O TransgenicAG consumiu em média 21,94%, 13,50% e 9,42% menos tempo de processamento nos experimentos com populações de 100, 200 e 1000 cromossomos. Por fim, no tempo total de execução, o TransgenicAG consumiu 26,28%, 24,34% e 72,48% menos tempo de processamento do que o StandardAG.

A única métrica que os resultados do StandardAG tiveram vantagem foi o número médio de gerações necessárias para convergir. O StandardAG apresentou resultados melhores do que o TransgenicAG com populações menores (populações de 100 e 200 cromossomos). Os melhores resultados do StandardAG apresentaram um ganho de 48,10% e 25,05% em comparação aos melhores resultados do TransgenicAG nas populações de 100 e 200 cromossomos, respectivamente. Contudo, com população de 1000 cromossomos, o TransgenicAG apresentou um desempenho 12,16% melhor do que o StandardAG.

Assim como ocorreu na função de *Beale*, na função de *McCormick*, um ganho apresentado pelos resultados do Algoritmo Transgênico em relação ao AG Convencional é o aumento das execuções que convergiram. A Figura 20 retrata graficamente os resultados. Nos experimentos com 100 cromossomos, o TransgenicAG atingiu a meta de convergência 10 vezes, enquanto que o StandardAG atingiu apenas 2 vezes. Com 200 cromossomos, o melhor resultado do TransgenicAG foi com mutação de 3%, onde convergiu 20 vezes, enquanto que o melhor resultado do StandardAG foram 5 convergências, tanto com 3% e 5% de probabilidade de mutação. Com 1000 cromossomos, o melhor resultado do TransgenicAG convergiu 89 vezes (nesse caso, com taxa de mutação de 5%), enquanto que o StandardAG convergiu 59 vezes. Ao aumentar a quantidade de cromossomos, pode-se observar o aumento considerável na quantidade de execuções convergentes.

5.4.2.2 Comparativo por taxa de mutação

Assim como na função *Beale*, os resultados da função *McCormick* não foram tão impactados pela taxa de mutação. Os resultados da maioria das métricas de avaliação sofreram pouca alteração ao utilizar uma taxa de mutação maior (de 3% para 5%). De maneira similar à função de *Beale*, a única métrica de avaliação que apresentou alterações mais significativas nos resultados ao mudar a taxa de mutação, foi o tempo total de execução. A Figura 21 apresenta o tempo total de execução do StandardAG e do TransgenicAG com as taxas de mutação de 3% e 5%. Como pode ser observado, apenas os experimentos com 1000 cromossomos apresentaram um ganho mais significativo na redução do tempo de execução. Com 1000 cromossomos, os experimentos com o TransgenicAG apresentaram uma redução considerável no tempo de execução em relação aos experimentos com

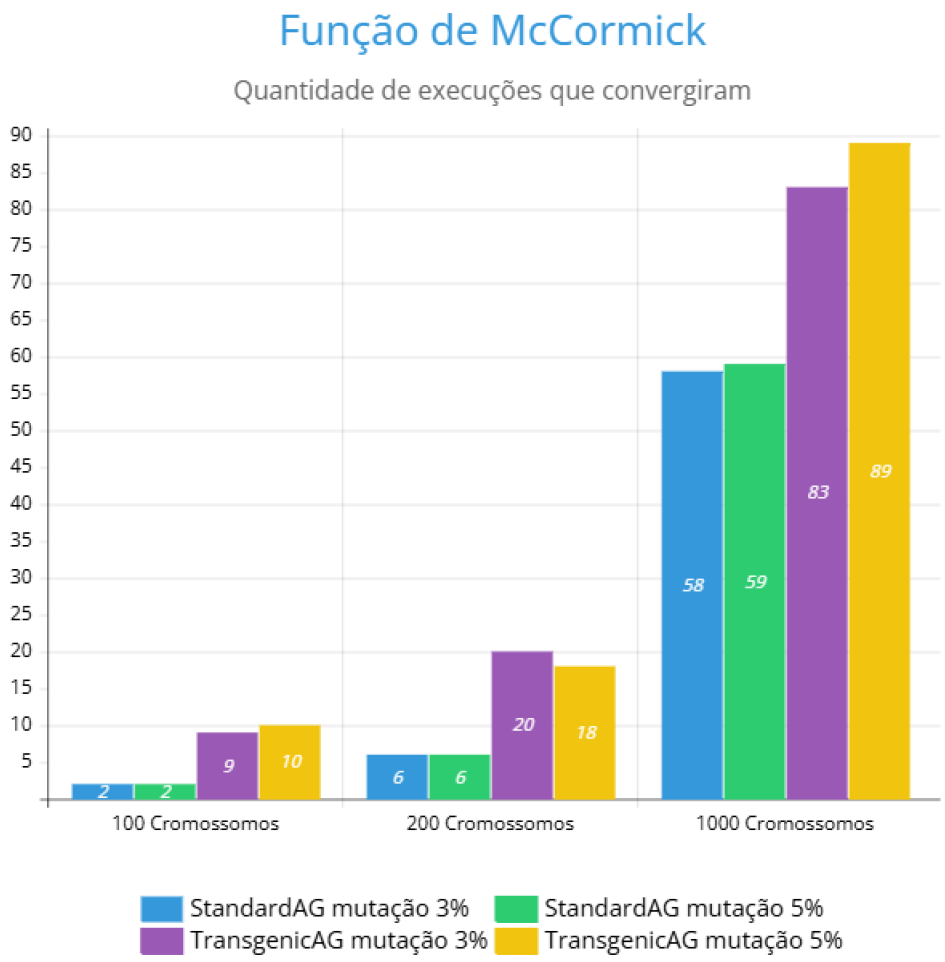


Figura 20 – Quantidade de execuções que encontram a solução ótima - Função McCormick

Fonte: elaborado pelo autor

o StandardAG. A redução ainda é maior ao utilizar a taxa de 5% ao invés de 3%. O TransgenicAG com mutação de 5% reduziu em 29,31% o tempo de execução em relação aos TransgenicAG com mutação de 3%. O StandardAG também apresentou redução no tempo de execução ao aumentar a taxa de 3% para 5%, porém foi uma redução tímida, apenas 5,43% menor.

5.4.3 Função Michalewicz

5.4.3.1 Comparativo entre o StandardAG x TransgenicAG

A função *Michalewicz* é única entre as funções selecionadas para este experimento. Ela possui um número variável de variáveis de entrada. O aumento do número de variáveis de entrada aumenta muito a complexidade da função. Testes foram realizados com três e quatro variáveis, mas nem mesmo uma única execução convergente foi obtida (mesmo utilizando populações de mil cromossomos não houve resultados que convergiram).

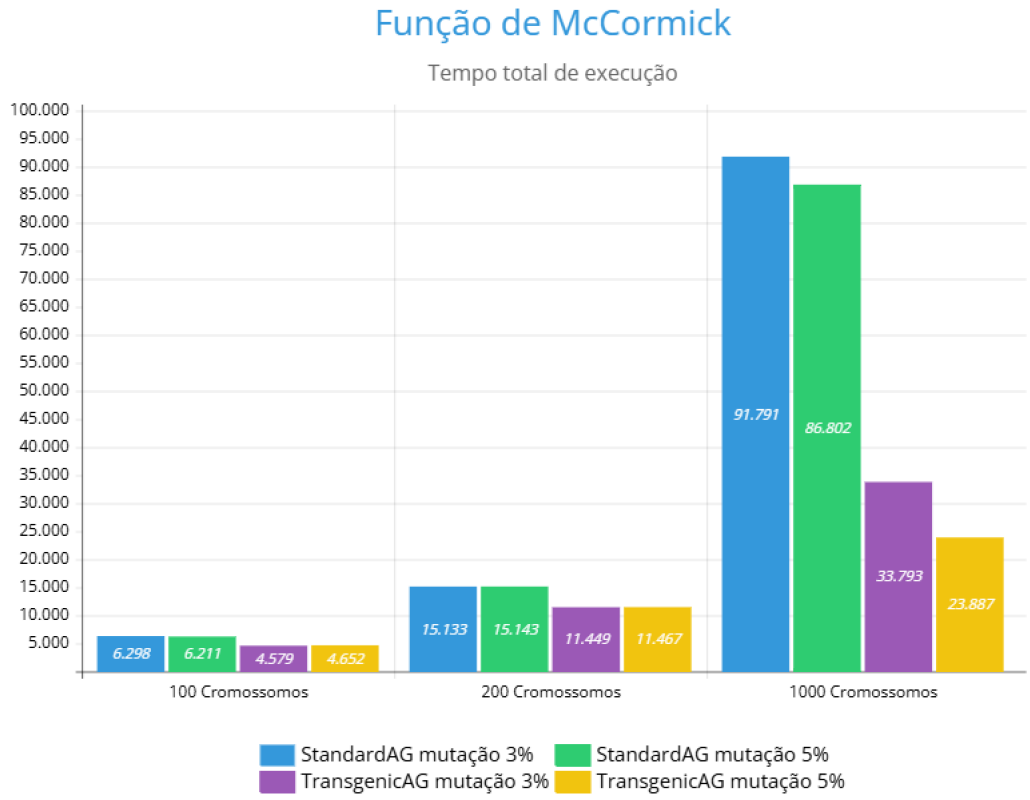


Figura 21 – Tempo total de execução da função *McCormick*

Fonte: elaborado pelo autor

Para otimização da função *Michalewicz* com mais de duas variáveis de entrada, é necessário um aumento significativo no número de cromossomos, talvez, além de se exigir alterações em outros parâmetros de inicialização do AG. Portanto, experimentos futuros com diferentes configurações serão realizados com a função *Michalewicz* para avaliar seu desempenho com mais de duas variáveis de entrada. Desse modo, todos os resultados apresentados neste experimento da função *Michalewicz* são realizados com duas variáveis.

Em relação à quantidade de execuções que encontraram o valor ótimo, a função Michalewicz utilizando duas variáveis obteve resultados consistentes. A Figura 22 ilustra graficamente os resultados. Nas populações de 100 e 200 cromossomos, o TransgenicAG convergiu mais vezes em comparação ao StandardAG. O TransgenicAG convergiu 64 e 92 vezes, enquanto o StandardAG convergiu 44 e 72 vezes respectivamente. Nos experimentos com população de 1000 cromossomos, ambos AGs obtiveram o desempenho máximo, convergindo em todas as execuções.

Analisando o tempo total de execução, o TransgenicAG foi mais rápido do que o StandardAG em todos os experimentos, apresentando resultados significativamente melhores nas populações de 100 e 200 cromossomos. O TransgenicAG foi mais rápido 50,20%, 66,59% e 26,59% do que o StandardAG nas populações de 100, 200 e 1000 cromossomos.

Diferentemente das outras duas funções, o TransgenicAG convergiu com menos gera-

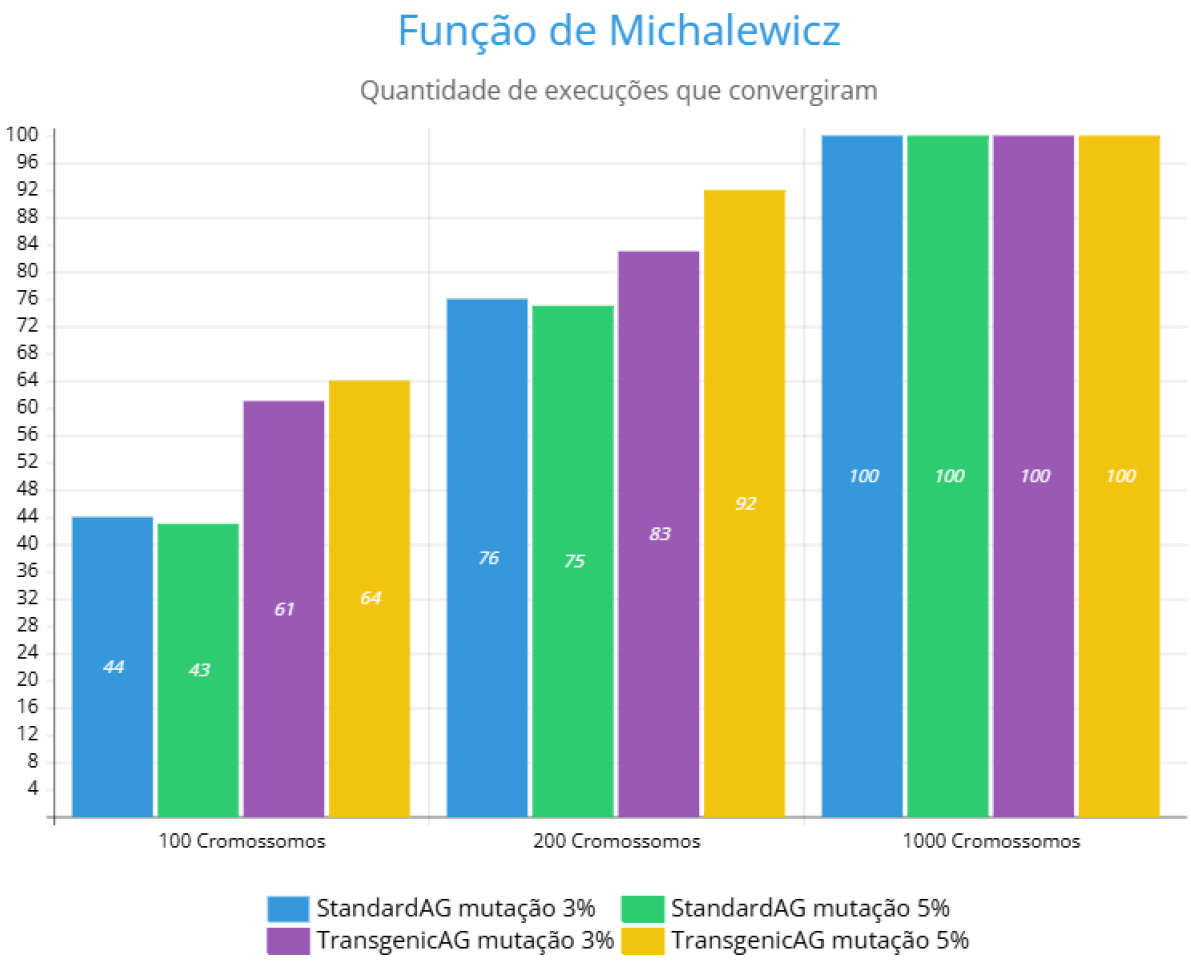


Figura 22 – Quantidade de execuções que convergiram na Função *Michalewicz*

Fonte: elaborado pelo autor

ções em média do que o StandardAG em todas as populações. A vantagem foi de 9,66%, 14,32% e 8,23% menos gerações em média para o TransgenicAG nas populações de 100, 200 e 1000 cromossomos em relação ao StandardAG.

O TransgenicAG demonstrou resultados significativamente melhores em comparação ao StandardAG em relação à média de tempo das execuções que convergiram. Os experimentos com população de 100 cromossomos obtiveram os maiores ganhos; o melhor resultado do Algoritmo Genético Transgênico foi 50,21% mais rápido do que o Algoritmo Genético convencional. Na população de 200 cromossomos, a vantagem foi de 26,41%. Enquanto na população de 1000 cromossomos, o melhor resultado do Algoritmo Genético Transgênico foi 25,09% mais rápido do que o melhor resultado do Algoritmo Genético convencional.

Por fim, o TransgenicAG também obteve melhores resultados do que o StandardAG no tempo médio das execuções que não convergiram. O melhor resultado do TransgenicAG foi 24,18% e 16,66% mais rápido que o melhor resultado do TransgenicAG nas

populações de 100 e 200 cromossomos, respectivamente. Os resultados com população de 1000 cromossomos não foram mensurados, porque houve 100% de sucesso de convergência em todas as execuções do StandardAG e do TransgenicAG com população de 1000 cromossomos. Desse modo, não houve execuções que não convergiram com esta quantidade populacional.

5.4.3.2 Comparativo por taxa de mutação

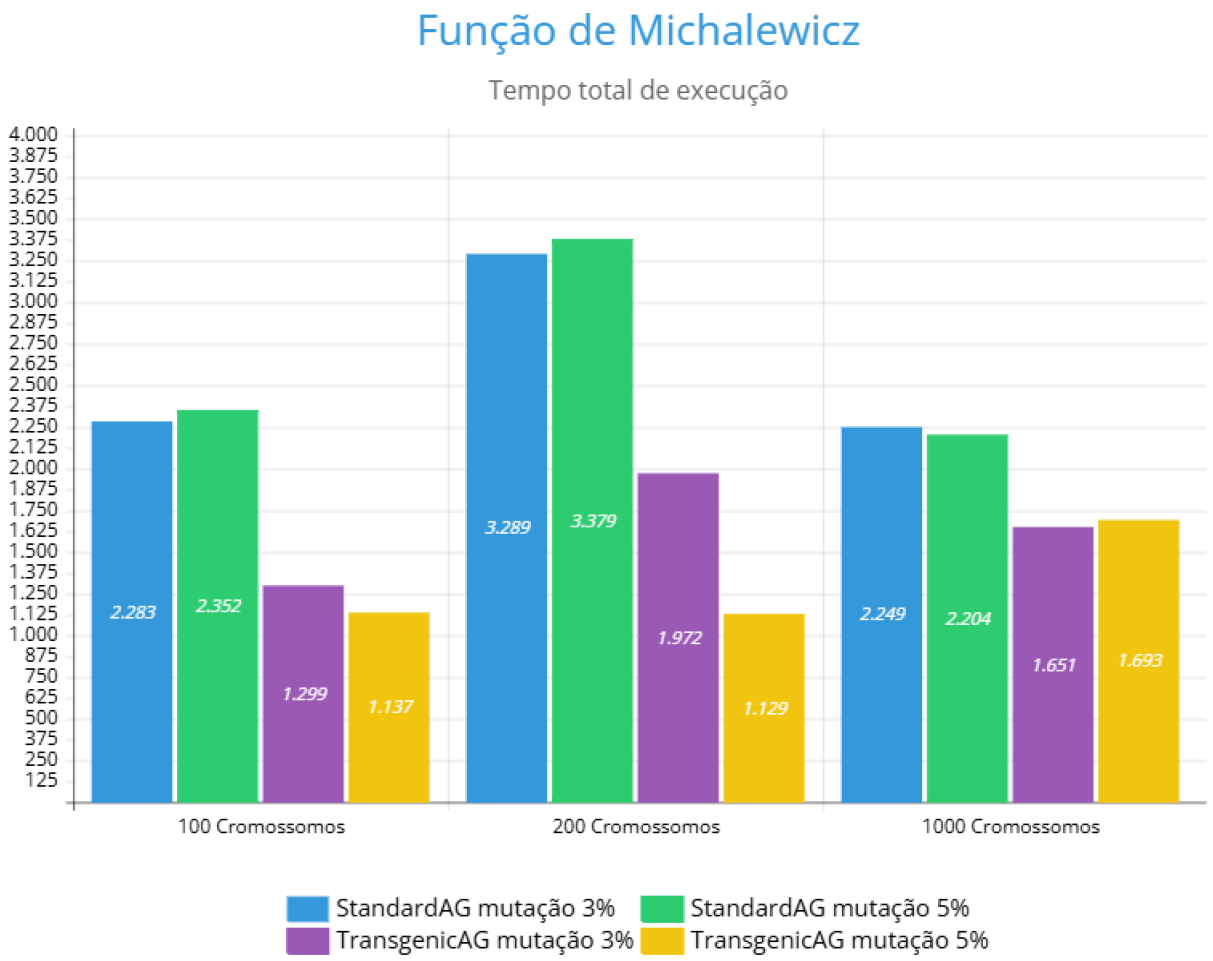


Figura 23 – Tempo total de execução da função da Função Michalewicz

Fonte: elaborado pelo autor

Assim como nas outras funções matemáticas, a alteração na taxa de mutação não proporcionou uma grande alteração aos resultados. O único resultado onde o aumento da taxa de mutação de 3% para 5% produziu um ganho significativo foi no tempo total de execução com 200 cromossomos. A Figura 23 apresenta o tempo total de execução dos experimentos da função *Michalewicz*. Como pode ser observado visualmente pela Figura 23, apenas os experimentos com 200 cromossomos apresentaram redução mais impactante no tempo de execução. O uso da taxa de mutação de 5% reduziu em 42,75% o tempo de

execução da função *Michalewicz* em relação à taxa de mutação de 3%. Nos experimentos com 100 cromossomos, a redução foi de 12,47%, enquanto com 1000 cromossomos ocorreu o inverso, onde a taxa de mutação de 3% apresentou maior redução. Nesse cenário, a taxa de mutação de 3% apresentou um ganho de 2,48% em relação à taxa de mutação de 5%.

Capítulo 6

Conclusão

Este trabalho começou a partir de uma proposta com vários objetivos e desafios. Entre os objetivos e desafios alcançados, este trabalho promoveu a implementação de um Algoritmo Genético convencional (StandardAG) e de um Algoritmo Genético Transgênico (TransgenicAG) baseado no operador *transgenic* de Amaral e Hruschka (2011), especialmente direcionados à otimização de fórmulas matemáticas. O Algoritmo Genético Transgênico é uma variação do Algoritmo Genético convencional desenvolvido com o objetivo de atingir o resultado ideal (ótimos globais) com o menor número de gerações possíveis. Além disso, a proposta deste trabalho considera um estudo comparativo e analítico entre o Algoritmo Genético Transgênico e o Algoritmo Genético convencional na otimização de fórmulas matemáticas. Muitos testes com diferentes funções matemáticas foram aplicados para avaliar as vantagens e desvantagens de ambas as abordagens. Os resultados dos experimentos possibilitaram conclusões importantes, as quais são destacadas neste capítulo.

6.1 Principais Contribuições

Neste trabalho, foram desenvolvidas duas diferentes abordagens de Algoritmos Genéticos para otimizar e analisar os impactos da otimização de problemas matemáticos com múltiplos ótimos locais. Ambos os métodos acrescentaram novas perspectivas às técnicas de otimização vigentes e permitiram a realização de uma investigação sobre os benefícios do uso e não uso da transgenia durante a otimização.

Somado a isso, a pesquisa realizada durante este trabalho possibilitou a validação das hipóteses levantadas na Seção 1.3, as quais trazem importantes contribuições. Segue abaixo cada hipótese e sua respectiva resposta:

- O operador de transgenia aumenta a capacidade de convergência de um Algoritmo Genético?

  Em todos os experimentos realizados, a adição do operador *transgenic* no Algoritmo Genético (TransgenicAG) aumentou significativamente a convergência. Independentemente da função matemática otimizada, da quantidade populacional e da taxa de mutação utilizada, o Algoritmo Transgênico convergiu mais vezes do que o Algoritmo Genético Convencional em todos os cenários de experimentos.

- Em relação ao número de gerações, o Algoritmo Genético Transgênico é mais eficiente, ou seja, converge com menos gerações, do que um Algoritmo Genético convencional?

  Não. Os resultados dos experimentos demonstraram que o Algoritmo Genético convencional necessitou de menos gerações para encontrar a solução ótima do que o Algoritmo Genético Transgênico em seis dos nove cenários de experimentos realizados (66,66% das oportunidades). Porém, deve-se levar em conta que o Algoritmo Transgênico convergiu mais vezes do que o Algoritmo Genético convencional. Assim, nas execuções onde o AG transgênico convergiu e o convencional não, geralmente ele necessitou de mais gerações para convergir do que as execuções onde os dois convergiram. O cálculo dessa métrica considera apenas as execuções que convergiram, assim as execuções tanto do convencional quanto do transgênico que atingiram o número máximo de gerações populacionais são desconsiderados no cálculo. Portanto, por convergir mais vezes, o cálculo da média de gerações necessárias para encontrar a solução ótima do AG Transgênico acaba sendo impactada. OS três cenários onde o Algoritmo Transgênico encontrou o valor ótimo com menos gerações populacionais em média, foram com a função *Michalewicz* em todas as populações utilizadas.

- Em relação ao tempo de processamento, o Algoritmo Genético Transgênico converge em menos tempo que um Algoritmo Genético convencional?

  Sim. Em todos os cenários, o AG transgênico consumiu menos tempo de processamento do que o AG convencional quando encontrou a solução ótima. Em 100% dos experimentos realizados, considerando apenas as execuções que convergiram, o AG transgênico apresentou um ganho computacional relacionado ao tempo de execução comparado ao AG convencional, demonstrando o potencial da transgenia na redução do tempo de convergência. Na maioria dos experimentos (mais precisamente em seis dos nove experimentos), o Algoritmo Transgênico com taxa de mutação de 5% obteve os melhores resultados.

- Quando o AG não converge para um valor ótimo, o Algoritmo Genético Transgênico consome menos tempo de processamento que o AG convencional?

Sim. Analisando somente as execuções que não encontram a solução ótima, o AG transgênico teve menor tempo médio de processamento do que o AG convencional em seis de oito experimentos realizados (75% dos casos). Nos demais experimentos, o AG convencional obteve uma média menor de tempo de execução.

6.2 Trabalhos Futuros

O estudo realizado por este trabalho promoveu a implementação de um Algoritmo Genético convencional (StandardAG) e de um Algoritmo Genético Transgênico (TransgenicAG) baseado no operador *transgenic* de Amaral e Hruschka (2011), especialmente direcionados à otimização de fórmulas matemáticas. Os resultados obtidos pelos experimentos utilizando o TransgenicAG e o StandardAG permitiram a validação das hipóteses levantadas, mas além disso, permitiram um aprofundamento e levantamento de novas conjecturas, que necessitam de validação, motivando a realização de novas pesquisas relacionadas ao tema.

Os experimentos evidenciaram impactos positivos do uso da transgenia. A transgenia resultou em um aumento na convergência, além de redução no tempo de processamento total, tempo de processamento das execuções que convergiram e no tempo de processamento das execuções que não convergiram. No entanto, mais testes são necessários para afirmar uma conclusão sólida sobre estes impactos. Experimentos adicionais são necessários para identificar outros impactos relacionados à qualidade, precisão e eficiência do Algoritmo Genético Transgênico. A exceção do número populacional e da taxa de mutação, todos os experimentos realizados neste estudo utilizaram os mesmos parâmetros para o Algoritmo Genético convencional e o Algoritmo Genético Transgênico. Um trabalho futuro utilizando diferentes configurações dos parâmetros de entrada para ambos os algoritmos, é previsto e está planejado. O intuito é verificar a reação de ambos os algoritmos e os impactos causados por cada mudança na parametrização.

O estudo futuro também considerará experimentos com populações substancialmente maiores do que as utilizadas nos experimentos. A função *Michalewicz* utilizada nos experimentos possui complexidade de otimização variável. Ao aumentar a quantidade de valores de entrada da função *Michalewicz*, ela aumenta a sua complexidade. Populações de 1000 cromossomos são insuficientes para otimizá-la com múltiplos valores de entrada. Portanto, o estudo futuro contemplará além de experimentos com diferentes configurações de parâmetros, a utilização de populações substancialmente maiores na tentativa de otimização de funções matemáticas mais complexas.

Os experimentos dessa pesquisa utilizaram taxas de mutação de 3% e 5%. O estudo futuro considera também a realização de experimentos com altas taxas de mutação. Amaral e Hruschka (2013) no seu trabalho *Transgenic: An Evolutionary Algorithm Operator*, identificaram que o operador *transgenic* exerce uma forte pressão seletiva. Essa caracte-

rística pode trazer benefícios como a perpetuação dos melhores genes entre as populações, o que facilitaria a otimização. Contudo, a forte pressão seletiva pode ocasionar problemas de estacionamento em ótimos locais, o que pode estagnar a busca do AG pela solução ótima em alguns casos. A mutação seria o fator chave para contornar o problema de estacionamento em ótimos locais. Assim, um estudo mais aprofundado a respeito do uso do Algoritmo Genético Transgênico com altas taxas de mutação na otimização de problemas matemáticos, principalmente problemas mais complexos com múltiplos picos locais, é fundamental para a descoberta de novos impactos do uso do operador *transgenic*.

6.3 Contribuições em Produção Bibliográfica

Até o presente momento, através dos conhecimentos e resultados obtidos pela pesquisa realizada neste trabalho, foi produzido um artigo científico que foi aprovado e publicado em 2025 no Congress on Evolutionary Computation (CEC). O artigo intitulado *Impacts of the Transgenic Operator on Genetic Algorithm Optimization of Mathematical Functions* argumenta sobre as vantagens e as desvantagens do uso do operador transgênico na otimização de funções matemáticas. Como é possível observar, tanto o acervo bibliográfico quanto a proposta, os objetivos, hipóteses e principalmente o método e os experimentos utilizados nesta pesquisa serviram de influência direta na elaboração deste artigo científico.

Referências

ALBERTS, Bruce (Ed.). **Molecular biology of the cell**. 4th ed. ed. New York: Garland Science, 2002. ISBN 978-0-8153-3218-3 978-0-8153-4072-0.

ALVES, Alexandre Henrick Da Silva; MENDES, Raphael De Lima; GOMES, Matheus De Souza; BERTARINI, Pedro Luiz Lima; AMARAL, Laurence Rodrigues Do. IG-CEE: An Embedded Information Gain approach to Genetic Algorithms. In: **2021 IEEE Congress on Evolutionary Computation (CEC)**. [S.l.]: IEEE, 2021. p. 1086–1092. ISBN 978-1-72818-393-0.

ALVES, Alexandre Henrick Da Silva; NETO, Guilherme Antonio Coelho; GOMES, Matheus De Souza; SANTOS, Líbia Diniz; BERTARINI, Pedro Luiz Lima; AMARAL, Laurence Rodrigues Do. Binary or Integer Chromosome: Which Is the Best Structure for Supervised Machine Learning Using Genetic Algorithms? **Applied Sciences**, v. 15, n. 5, p. 2608, fev. 2025.

AMARAL, Laurence Rodrigues Do; HRUSCHKA, Estevam Rafael. Transgenic, an operator for evolutionary algorithms. In: **2011 IEEE Congress of Evolutionary Computation (CEC)**. New Orleans, LA: IEEE, 2011. p. 1308–1314. ISBN 978-1-4244-7834-7 978-1-4244-7835-4. Disponível em: <https://ieeexplore.ieee.org/document/5949767/>.

AMARAL, Laurence Rodrigues Do; HRUSCHKA, Estevam Rafael. Transgenic: An evolutionary algorithm operator. **Neurocomputing**, v. 127, p. 104–113, mar. 2013. ISSN 09252312. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0925231213009375>.

AMARAL, Laurence Rodrigues do; OLIVEIRA, Gina Maira Barbosa de. Mineração de regras para classificação de oncogenes medidos por microarray utilizando algoritmos genéticos. **Revista Brasileira de Cancerologia**, v. 56, n. 3, p. 391, jun. 2010.

AMARAL, Laurence Rodrigues do; HRUSCHKA, Estevam Rafael. Gene ontology classification: Building high-level knowledge using genetic algorithms. In: **IEEE Congress on Evolutionary Computation**. [S.l.: s.n.], 2010. p. 1–7.

ARORA, Geeta; SAHLOT, Simran; BALA, Kiran. MATHEMATICS Exploring Optimization Technique: Genetic Algorithm. **SSRN Electronic Journal**, 2024. ISSN 1556-5068.

AVILA, Sérgio Luciano. **Otimização paramétrica com computação evolutiva**. [S.l.]: Publicação do IFSC, 2020. ISBN 9786588663004.

AZEVEDO, João Lúcio De; FUNGARO, Maria Helena Pelegrinelli; VIEIRA, Maria Lúcia Carneiro. Transgênicos e evolução dirigida. **História, Ciências, Saúde-Manguinhos**, v. 7, n. 2, p. 451–464, out. 2000. ISSN 0104-5970. Disponível em: <https://www.scielo.br/j/hcsm/a/DVzFRf5jZXrQTkxV6LrLpBy/?lang=pt>.

BINGHAM, Derek; SURJANOVIC, Sonja. **Virtual Library of Simulation Experiments**. Simon Fraser University, 2013. Disponível em: <https://www.sfu.ca/~ssurjano/optimization.html>.

BOEIRA, Julia Naomi Rosenfield. The Effects of "Preferentialism" on a Genetic Algorithm Population over Elitism and Regular Development in a Binary F6 Fitness Function. **International Journal of Intelligent Systems and Applications**, v. 8, n. 9, p. 38–46, set. 2016.

BQIBR. **Composição da Biotecnologia**. 2017. Disponível em: <https://bioquimicabrasil.com/2017/07/02/diferencas-bioquimica-biotecnologia/>.

BUENO, Fabrício. Métodos heurísticos: Teoria e implementações. **IFSC**, n. 42, 2009.

CUNHA, Lucas; FERREIRA, Luciana; RODRIGUES, Fabrício Alves; ALENCAR, Wanderley. **Lógica para Semáforo Inteligente Baseado na Mineração de Dados por Algoritmo Genético Transgênico**. 2013.

DARWIN, Charles. **On the Origin of Species by Means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life.** [S.l.: s.n.], 1859.

DAVIS, L. **Handbook of Genetic Algorithms**. [S.l.]: International Thomson Computer Press, 1996. ISBN 9781850328254.

DEEP, Kusum; SINGH, Krishna Pratap; KANSAL, M.L.; MOHAN, C. A real coded genetic algorithm for solving integer and mixed integer optimization problems. **Applied Mathematics and Computation**, v. 212, n. 2, p. 505–518, jun. 2009. ISSN 00963003.

ERTEL, Wolfgang. **Introduction to Artificial Intelligence**. [S.l.]: Springer International Publishing, 2017. (Undergraduate Topics in Computer Science). ISBN 978-3-319-58487-4.

FILLITO, Danilo. Algoritmos genéticos: uma visão explanatória. **Revista Multidisciplinar da UNIESP**, 2008.

FOGEL, David B. The advantages of evolutionary computation. **Biocomputing and Emergent Computation**, 1997.

GEN, Mitsuo; TSUJIMURA, Yasuhiro; KUBOTA, Erika. Solving job-shop scheduling problems by genetic algorithm. In: IEEE. **Proceedings of IEEE international conference on systems, man and cybernetics**. [S.l.], 1994. v. 2, p. 1577–1582.

GOLDBERG, David E. **Genetic Algorithms in Search, Optimization and Machine Learning**. 1st. ed. [S.l.]: Addison-Wesley Longman Publishing Co., Inc., 1989. ISBN 0201157675.

HOLLAND, John H. **Adaptation in Natural and Artificial Systems**. Ann Arbor, MI: University of Michigan Press, 1975. Second edition, 1992.

KATO, E. R. R.; MORANDIN, O.; FONSECA, M. A. S. Ant colony optimization algorithm for reactive production scheduling problem in the job shop system. In: **2009 IEEE International Conference on Systems, Man and Cybernetics**. [S.l.: s.n.], 2009. p. 2199–2204.

LIM, Siew Mooi; SULTAN, Abu Bakar Md.; SULAIMAN, Md. Nasir; MUSTAPHA, Aida; LEONG, K. Y. Crossover and Mutation Operators of Genetic Algorithms. **International Journal of Machine Learning and Computing**, v. 7, n. 1, p. 9–12, fev. 2017. ISSN 20103700.

MAN, Kim.F.; TANG, Kit.S.; KWONG, Sam. Genetic algorithms: concepts and applications. **IEEE Transactions on Industrial Electronics**, v. 43, n. 5, p. 519–534, out. 1996. ISSN 02780046.

MANUALDABIOLOGIA. **Transgenia – Engenharia Genética**. 2024. Disponível em: <https://www.manualdabiologia.com.br/2024/12/transgenia-engenharia-genetica-omg.html>.

MATOS, Maicon Douglas Santos; AMARAL, Laurence Rodrigues Do. **Multiple Disjunctions Rule Genetic Algorithm (MDRGA): Inferring Non-Linear IF-THEN Rules in Non-Linear Datasets**. 2018. 1-6 p.

MITCHELL, Melanie. **An introduction to genetic algorithms**. Cambridge, Mass: MIT Press, 1996. (Complex adaptive systems). ISBN 978-0-262-13316-6.

MORANDIN, Orides; KATO, Edilson R. R.; DERIZ, Ana C.; SANCHES, Danilo S. A search method using genetic algorithm for production reactive scheduling of manufacturing systems. In: **2008 IEEE International Symposium on Industrial Electronics**. [S.l.: s.n.], 2008. p. 1843–1848.

MORANDIN, O.; SANCHES, D. S.; DERIZ, A. C.; KATO, E. R. R.; TSUNAKI, R. H. An adaptive genetic algorithm based approach for production reactive scheduling of manufacturing systems. In: **2008 34th Annual Conference of IEEE Industrial Electronics**. [S.l.: s.n.], 2008. p. 1461–1466.

OLIVEIRA, Gina M. B.; SILVA, Reslley G. O.; FERREIRA, Giordano B. S.; COUCEIRO, Micael S.; AMARAL, Laurence R. do; VARGAS, Patricia A.; MARTINS, Luiz Gustavo A. **A Cellular Automata-Based Path-Planning for a Cooperative and Decentralized Team of Robots**. 2019. 739-746 p.

PACHECO, Marco Aurélio Cavalcanti *et al.* Algoritmos genéticos: princípios e aplicações. **ICA: Laboratório de Inteligência Computacional Aplicada. Departamento de Engenharia Elétrica. Pontifícia Universidade Católica do Rio de Janeiro.**, v. 28, 1999.

PILA, Adriano Donizete. História e terminologia a respeito da computação evolutiva. **Revista de Ciências Exatas e Tecnologia 1**, p. 40–50, 2006.

PIMENTEL, D.; HUNTER, M. S.; LAGRO, J. A.; EFROYMSON, R. A.; LANDERS, J. C.; MERVIS, F. T.; MCCARTHY, C. A.; BOYD, A. E. Benefits and Risks of Genetic Engineering in Agriculture. **BioScience**, v. 39, n. 9, p. 606–614, out. 1989.

PINTO, A. R. F.; MARTARELLI, N. J.; NAGANO, M. S. Algoritmo Genético: Principais Gaps, Trade-offs e Perspectivas para Futuras Pesquisas. **Trends in Computational and Applied Mathematics**, v. 23, n. 3, p. 413–438, set. 2022. ISSN 2676-0029.

REEVES, Colin. A genetic algorithm for flowshop sequencing. **Computers & Operations Research**, v. 22, n. 1, p. 5–13, 1995.

RODRIGUES, Fabrício Alves; SOUZA, Stéfanne; ARANHA, Renan; FREITAS, Ana. Application programming interface library para algoritmo genético com operador transgênico. In: . [S.l.: s.n.], 2013.

SASTRY, Kumara; GOLDBERG, David; KENDALL, Graham. **Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques**. Boston, MA: Springer US, 2005. 97–125 p. ISBN 978-0-387-28356-2.

SCHUBERT, Ingo. Chromosome evolution. **Current Opinion in Plant Biology**, v. 10, n. 2, p. 109–115, abr. 2007. ISSN 13695266.

SILVA, Jennifer Amanda Sobral da; MAIRINK, Carlos Henrique Passos. Inteligência artificial. **LIBERTAS: Revista de Ciênciais Sociais Aplicadas**, v. 9, n. 2, p. 64–85, 2019.

SILVA, Reslley Gabriel Oliveira; RODRIGUES, Fabrício Alves; BEVILAQUA, André; BARRETO, Franciny Medeiros; NASCIMENTO, Thamer Horbylon; AMARAL, Laurence Rodrigues do. **Construção de Conhecimento de Alto Nıvel a Partir de Datasets Biológicos com Alta Dimensionalidade Utilizando Algoritmos Genéticos**. 2011.

SPEARS, William M. Crossover or Mutation? In: **Foundations of Genetic Algorithms**. [S.l.]: Elsevier, 1993. v. 2, p. 221–237. ISBN 978-0-08-094832-4.

VARGAS, Bruna Damaceno; BASSO, Arthur; RODRIGUES, Thais Valença; SILVA, Luana Bizzi; GATZKE, Monica; FRIZZO, Matias Nunes. BIOTECNOLOGIA E ALIMENTOS GENETICAMENTE MODIFICADOS: UMA REVISÃO. **Revista Contexto & Saúde**, v. 18, n. 35, p. 19–26, dez. 2018. ISSN 2176-7114, 1676-188X. Disponível em: <https://www.revistas.unijui.edu.br/index.php/contextoesaude/article/view/5591>.

VIANA, Monique; JUNIOR, Orides Morandin; CONTRERAS, Rodrigo. Transgenic Genetic Algorithm to Minimize the Makespan in the Job Shop Scheduling Problem:. In: **Proceedings of the 12th International Conference on Agents and Artificial Intelligence**. [S.l.]: SCITEPRESS - Science and Technology Publications, 2020. p. 463–474.

VIANA, Monique Simplicio; CONTRERAS, Rodrigo Colnago; JUNIOR, Orides Morandin. A New Frequency Analysis Operator for Population Improvement in Genetic Algorithms to Solve the Job Shop Scheduling Problem. **Sensors**, v. 22, n. 12, p. 4561, jun. 2022.

WATSON, James D. (Ed.). **Molecular biology of the gene**. 7. ed., international ed. ed. Boston: Pearson, 2014. (Always learning). ISBN 978-0-321-85149-9.

WATSON, J. D.; CRICK, F. H. C. Molecular Structure of Nucleic Acids: A Structure for Deoxyribose Nucleic Acid. **Nature**, v. 171, n. 4356, p. 737–738, abr. 1953. ISSN 0028-0836, 1476-4687. Disponível em: <https://www.nature.com/articles/171737a0>.

WOODRUFF, Ronny C.; THOMPSON, James N. **Mutation and Evolution**. Dordrecht: Springer Netherlands, 1998. v. 7. (Contemporary Issues in Genetics and Evolution, v. 7). Disponível em: <http://link.springer.com/10.1007/978-94-011-5210-5>.

XAVIER, E. G.; LOPES, D. C. N.; PETERS, M. D. P. Organismos geneticamente modificados. **Archivos de Zootecnia**, v. 58, n. 224, p. 15–33, set. 2008. ISSN 1885-4494. Disponível em: <https://www.uco.es/ucopress/az/index.php/az/article/view/5072>.

Capítulo 7

Apêndice(s)

Neste apêndice são apresentados os resultados individuais dos experimentos realizados com os algoritmos StandardAG e TransgenicAG. Para facilitar a visualização, comparação e análise, as informações foram agrupadas em tabelas por função matemática e tamanho da população (quantidade de cromossomos por população). As colunas mut 3% e mut 5% indicam a quantidade de gerações que o AG específico demorou para convergir na respectiva taxa de mutação (100 gerações quando não convergiu). Os valores das colunas tempo são valores em milissegundos. Segue abaixo as tabelas com os resultados dos experimentos.

7.1 Resultados dos experimentos

7.1.1 Beale - População 100 cromossomos

	StandardAG					**TransgenicAG**				
Seed	mut 3%	tempo		mut 5%	tempo	mut 3%	tempo		mut 5%	tempo
1	6	57		6	59	4	44		4	42
2	100	97		100	205	100	147		100	198
3	100	31		100	30	100	56		100	52
4	100	46		100	46	100	51		100	55
5	5	2		5	3	4	2		4	2
6	100	46		100	63	100	42		100	58
7	100	42		100	56	100	36		100	43
8	100	49		100	54	100	47		100	44
9	100	26		100	33	100	31		100	32
10	100	38		100	32	100	44		100	42
11	100	20		100	19	100	29		100	28
12	100	20		100	18	100	44		100	41

13	100	26		100	29		100	24		100	24
14	100	26		100	25		100	47		100	54
15	100	32		100	24		100	46		100	39
16	100	51		100	49		100	50		100	41
17	100	54		100	44		100	23		100	22
18	100	27		100	16		100	42		100	59
19	100	54		100	42		100	41		100	42
20	100	107		100	40		100	111		100	56
21	100	45		100	38		100	67		100	53
22	100	23		100	20		100	50		100	51
23	100	29		100	19		100	38		100	31
24	100	42		100	24		100	56		100	52
25	100	41		100	35		100	59		100	41
26	1	1		1	1		1	1		1	1
27	100	44		100	33		100	54		100	49
28	100	45		100	35		100	45		100	40
29	100	22		100	16		100	42		100	45
30	100	42		100	29		100	43		100	47
31	100	45		100	37		100	61		100	42
32	100	42		100	46		100	70		100	52
33	100	21		100	25		100	23		100	22
34	100	38		100	48		100	72		100	43
35	100	37		100	62		100	41		100	67
36	100	35		100	37		100	43		100	50
37	5	1		5	1		100	42		100	42
38	4	1		4	1		5	1		5	2
39	100	46		100	41		100	43		100	54
40	100	28		100	34		100	26		100	25
41	100	19		100	19		100	61		100	63
42	100	40		100	38		100	48		100	43
43	100	42		100	47		100	22		100	31
44	100	41		100	51		100	41		100	57
45	4	1		4	1		2	1		2	1
46	100	42		100	53		100	43		100	42
47	100	44		100	50		100	41		100	48
48	100	23		100	19		100	37		100	46
49	100	26		100	20		100	52		100	40

50	100	45		100	37		7	2		7	2
51	100	50		100	55		100	47		100	55
52	100	18		100	19		100	23		100	27
53	100	35		100	39		10	6		10	4
54	100	30		100	24		100	51		100	43
55	100	30		100	33		100	42		100	43
56	100	45		100	39		100	56		100	41
57	100	19		100	20		100	54		100	55
58	100	23		100	17		100	19		100	19
59	100	41		100	40		100	57		100	40
60	100	39		100	47		100	48		100	40
61	100	23		100	20		100	50		100	47
62	2	1		2	1		2	1		2	1
63	100	43		100	42		100	42		100	44
64	7	2		7	2		5	1		5	1
65	100	46		100	39		100	42		100	45
66	100	43		100	40		100	34		100	31
67	100	44		100	44		100	39		100	43
68	5	1		5	1		4	1		4	2
69	100	41		100	38		100	39		100	43
70	100	28		100	26		100	51		100	34
71	100	24		100	22		100	22		100	42
72	100	30		100	23		100	31		100	36
73	100	43		100	36		100	40		100	34
74	100	45		100	46		100	43		100	49
75	100	58		100	41		100	42		100	43
76	100	48		100	42		100	41		100	39
77	4	1		4	1		3	1		3	1
78	100	43		100	33		100	40		100	45
79	100	37		100	26		100	26		100	30
80	100	44		100	39		100	40		100	33
81	100	46		100	46		100	26		100	21
82	100	23		100	21		100	38		100	40
83	4	1		4	1		100	43		100	49
84	100	25		100	28		100	38		100	36
85	100	47		100	39		100	41		100	43
86	100	47		100	48		100	41		100	45

87	100	48		100	46		100	45		100	35
88	4	1		4	1		2	1		2	1
89	100	48		100	41		100	44		100	50
90	100	41		100	47		100	39		100	44
91	100	21		100	25		100	41		100	49
92	6	2		6	2		6	2		6	3
93	100	46		100	48		100	39		100	31
94	100	45		100	38		100	40		100	35
95	100	21		100	19		100	36		100	41
96	100	37		100	37		100	45		100	61
97	100	54		100	48		100	58		100	56
98	5	1		5	1		7	3		7	2
99	100	21		100	19		100	43		100	41
100	100	44		100	41		100	48		100	42

Tabela 6 – Beale - População 100 indivíduos

7.1.2 Beale - População 200 cromossomos

	StandardAG						**TransgenicAG**				
seed	mut 3%	tempo		mut 5%	tempo		mut 3%	tempo		mut 5%	tempo
1	9	7		9	7		100	60		100	74
2	100	257		11	11		100	89		9	7
3	100	191		100	185		100	135		100	127
4	7	9		7	6		9	7		9	9
5	3	2		3	6		3	2		3	1
6	100	79		100	88		100	151		100	122
7	100	113		100	100		100	78		100	95
8	10	8		10	7		7	6		7	4
9	100	177		100	175		100	160		100	161
10	100	119		100	163		100	142		100	128
11	100	110		100	105		100	137		100	114
12	100	131		100	164		100	127		100	120
13	100	132		100	149		100	152		100	164
14	4	3		4	3		7	9		7	6
15	100	53		100	55		100	56		100	42
16	100	110		100	120		100	133		100	135
17	100	109		100	104		100	108		100	94

18	100	126		100	137		100	127		100	125
19	10	13		10	8		100	165		100	155
20	100	161		100	190		100	174		100	194
21	100	60		100	61		100	56		100	78
22	2	2		2	1		2	1		2	2
23	100	165		100	156		100	132		100	136
24	100	160		100	158		100	163		100	170
25	5	6		5	3		100	111		100	103
26	1	1		1	1		1	1		1	1
27	100	86		100	91		100	55		100	67
28	5	3		5	4		5	4		5	3
29	100	94		100	99		100	90		100	95
30	100	1		100	2		4	2		4	2
31	4	2		4	2		3	5		3	3
32	100	136		100	137		3	2		3	2
33	100	132		100	133		100	142		100	137
34	5	6		5	5		100	163		100	169
35	100	137		100	73		100	165		100	127
36	100	47		100	46		100	62		100	73
37	100	55		100	54		100	94		100	105
38	3	2		3	2		5	3		5	2
39	100	115		100	130		100	102		100	107
40	5	2		5	2		3	4		3	3
41	3	2		3	3		4	3		4	1
42	100	134		100	175		100	154		100	152
43	100	142		100	138		100	149		100	141
44	3	2		3	4		5	2		5	2
45	9	8		9	8		5	4		5	4
46	100	73		100	75		100	75		100	73
47	100	106		100	108		100	125		100	127
48	10	9		10	7		3	4		3	4
49	100	56		100	55		100	47		100	47
50	100	53		100	52		100	77		100	65
51	100	154		100	145		100	153		100	138
52	100	122		100	126		100	115		100	120
53	8	6		8	9		4	6		4	4
54	4	3		4	2		3	3		3	2

55	100	54		100	49		100	67		100	52
56	100	56		100	59		100	71		100	79
57	100	47		100	52		100	70		100	74
58	8	6		8	7		5	4		5	2
59	8	7		8	4		4	3		4	5
60	100	3		100	6		6	5		6	7
61	3	1		3	3		3	2		3	1
62	5	2		5	6		5	2		5	3
63	100	84		100	93		100	61		100	64
64	2	1		2	1		2	2		2	1
65	8	6		8	8		6	5		6	4
66	100	77		100	79		100	103		100	128
67	100	131		100	136		100	142		100	157
68	2	2		2	1		2	2		2	2
69	4	6		4	4		9	6		9	9
70	2	1		2	2		2	2		2	1
71	100	147		100	145		100	153		100	146
72	100	65		100	51		100	62		100	96
73	100	135		100	141		100	151		100	153
74	100	83		100	92		7	6		7	5
75	100	140		100	131		100	131		100	139
76	3	2		3	2		6	3		6	4
77	4	3		4	2		3	1		3	3
78	100	154		100	134		100	161		100	69
79	100	105		100	110		100	105		100	109
80	100	101		100	116		100	111		100	118
81	100	83		100	70		100	56		100	56
82	100	98		100	122		100	84		100	85
83	100	122		100	124		100	77		100	96
84	100	50		100	61		100	141		100	133
85	100	149		100	160		100	153		100	163
86	100	80		100	83		100	71		100	85
87	2	1		2	1		2	1		2	1
88	7	8		7	6		7	8		7	2
89	100	137		100	144		100	99		100	108
90	100	105		100	110		100	133		100	144
91	100	140		100	124		100	146		100	146

92	100	83		100	5		6	4		6	4
93	100	118		100	85		100	69		100	70
94	100	137		100	148		100	147		100	162
95	100	75		100	87		100	64		100	70
96	100	126		100	130		100	64		100	61
97	100	89		100	92		100	71		100	85
98	100	83		100	89		100	64		100	63
99	4	3		4	3		5	6		5	5
100	4	2		4	3		5	3		5	4

Tabela 7 – Beale - População 200 indivíduos

7.1.3 Beale - População de 1000 cromossomos

	StandardAG						**TransgenicAG**			
Seed	mut 3%	tempo		mut 5%	tempo		mut 3%	tempo		mut 5%	tempo
1	3	148		3	154		3	124		3	135
2	4	55		2	59		5	61		2	68
3	2	11		2	20		2	13		2	48
4	2	10		2	26		2	9		2	21
5	3	20		3	31		3	12		3	27
6	2	11		2	15		2	15		2	18
7	8	143		8	126		8	111		8	124
8	100	1638		100	1591		100	1374		100	1406
9	5	30		5	37		3	19		3	18
10	2	8		2	10		2	13		2	10
11	2	10		2	11		2	9		2	8
12	6	40		6	43		5	27		5	31
13	2	9		2	10		2	8		2	9
14	3	17		3	17		4	25		4	31
15	6	52		6	48		6	41		6	42
16	5	31		5	36		4	26		4	30
17	6	33		6	46		3	20		3	16
18	10	63		10	78		7	56		7	49
19	5	31		5	36		4	26		4	22
20	5	64		5	68		4	66		4	57
21	4	31		4	33		4	30		4	33
22	5	41		5	42		4	29		4	30

23	2	15		2	16		2	9		2	13
24	4	30		4	34		4	35		4	23
25	1	4		1	4		1	4		1	4
26	1	4		1	4		1	4		1	3
27	4	28		4	33		3	16		3	20
28	4	32		4	34		3	19		3	15
29	5	45		5	39		5	38		5	32
30	6	44		6	53		6	46		6	41
31	3	19		3	21		6	40		6	42
32	3	16		3	19		100	1531		100	1586
33	100	1520		100	1427		100	2088		100	2155
34	5	48		5	58		4	21		4	22
35	3	16		4	46		4	22		6	58
36	4	20		4	45		3	15		3	17
37	6	35		6	50		5	27		5	30
38	3	15		3	21		3	13		3	14
39	1	4		1	3		1	3		1	2
40	5	29		5	33		5	26		5	37
41	1	3		1	3		1	2		1	3
42	5	27		5	31		4	20		4	21
43	2	11		2	7		2	9		2	9
44	2	9		2	9		2	8		2	9
45	5	52		5	35		7	40		7	44
46	2	7		2	10		2	7		2	8
47	5	27		5	37		6	33		6	33
48	2	7		2	9		2	8		2	11
49	4	21		4	29		4	19		4	21
50	2	8		2	14		2	8		2	8
51	2	9		2	9		2	8		2	11
52	5	25		5	44		3	18		3	14
53	2	12		2	15		2	28		2	12
54	2	14		2	15		2	13		2	8
55	2	12		2	11		2	11		2	8
56	7	72		7	50		7	55		7	40
57	5	49		5	35		4	28		4	21
58	6	58		6	50		6	53		6	37
59	3	18		3	25		4	19		4	21

60	4	26		4	31		3	13		3	15
61	2	15		2	9		2	17		2	11
62	2	12		2	11		2	4		2	12
63	4	30		4	29		4	25		4	26
64	7	84		7	75		5	40		5	31
65	4	25		4	26		4	22		4	20
66	9	68		9	63		3	21		3	14
67	4	26		4	24		5	28		5	27
68	100	1705		100	1755		7	45		7	40
69	1	2		1	4		1	3		1	4
70	5	62		5	64		6	38		6	31
71	1	3		1	2		1	3		1	2
72	1	3		1	2		1	4		1	2
73	5	28		5	27		6	53		6	32
74	1	2		1	2		1	5		1	2
75	4	21		4	22		10	35		10	37
76	6	34		6	35		100	3604		100	1409
77	2	7		2	7		2	7		2	5
78	3	16		100	2265		3	13		100	3453
79	5	28		5	24		5	26		5	27
80	11	111		11	81		8	42		8	45
81	6	42		6	47		100	2987		100	3148
82	100	2224		100	1495		100	2708		100	2348
83	100	2079		100	2119		7	43		7	42
84	7	43		7	42		4	22		4	22
85	3	15		3	17		3	14		3	14
86	8	47		8	56		6	37		6	33
87	3	15		3	35		3	13		3	14
88	2	8		2	19		2	7		2	8
89	5	79		5	71		6	53		6	55
90	6	34		6	29		4	18		4	21
91	4	21		4	53		3	11		3	17
92	3	14		3	33		3	15		3	14
93	4	21		4	48		3	10		3	13
94	2	8		2	13		2	10		2	8
95	4	21		4	23		5	25		5	19
96	100	2708		100	2691		100	2339		100	1624

97	4	27		4	38		4	25		4	25
98	4	26		4	54		5	28		5	26
99	100	792		100	844		100	859		100	905
100	3	15		3	19		5	31		5	25

Tabela 8 – Beale - População 1000 indivíduos

7.1.4 McCormick - População 100 cromossomos

	StandardAG					**TransgenicAG**				
Seed	mut 3%	tempo		mut 5%	tempo		mut 3%	tempo		mut 5%	tempo
1	100	279		100	250		100	216		100	208
2	100	198		100	281		100	128		100	157
3	100	134		100	136		100	109		100	111
4	100	112		100	93		100	71		100	94
5	100	93		100	60		100	74		100	60
6	100	83		100	85		100	76		100	71
7	100	80		100	77		100	89		100	82
8	100	69		100	52		100	90		100	60
9	100	64		100	73		100	45		100	55
10	100	59		100	57		100	68		100	56
11	100	68		100	82		100	63		100	69
12	100	71		100	87		100	31		100	45
13	100	54		100	50		100	62		100	59
14	100	66		100	65		100	65		100	66
15	100	57		100	56		100	47		100	43
16	100	57		100	54		100	63		100	53
17	100	66		100	82		100	57		100	55
18	100	53		100	53		100	70		100	63
19	100	63		100	47		100	73		100	63
20	100	70		100	99		100	50		100	67
21	100	63		100	69		100	81		100	60
22	100	53		100	55		100	65		100	56
23	100	58		100	54		100	59		100	66
24	100	63		100	58		100	67		100	60
25	100	71		100	79		100	46		100	48
26	100	64		100	55		100	58		100	64
27	100	60		100	52		100	66		100	56

28	100	36		100	30		100	46		100	40
29	3	3		3	4		2	1		2	2
30	100	57		100	58		100	77		100	62
31	100	56		100	62		100	56		100	63
32	100	52		100	53		100	56		100	59
33	100	60		100	51		100	56		100	57
34	100	57		100	51		100	58		100	61
35	100	60		100	54		100	55		100	43
36	100	65		100	54		100	55		100	57
37	100	59		100	49		100	55		100	59
38	100	56		100	68		100	57		100	56
39	100	57		100	47		100	72		100	59
40	100	57		100	55		100	54		100	69
41	100	57		100	53		100	53		100	58
42	100	59		100	50		100	71		100	62
43	100	60		100	52		100	76		100	61
44	100	65		100	69		100	58		100	57
45	100	66		100	52		100	59		100	58
46	100	41		100	46		100	61		100	52
47	100	62		100	52		100	62		100	59
48	100	63		100	54		100	61		100	57
49	100	79		100	88		100	100		100	87
50	100	34		100	36		100	42		100	46
51	100	70		100	78		100	76		100	73
52	100	52		100	53		100	72		100	75
53	3	1		3	1		3	2		3	1
54	100	52		100	55		100	50		100	56
55	100	59		100	49		100	43		100	47
56	100	38		100	34		100	45		100	43
57	100	56		100	58		100	52		100	59
58	100	60		100	54		100	53		100	51
59	100	61		100	58		100	56		100	52
60	100	62		100	51		100	66		100	60
61	100	57		100	59		100	65		100	61
62	100	57		100	56		100	43		100	55
63	100	62		100	52		100	56		100	63
64	100	47		100	44		100	64		100	57

65	100	50		100	51		100	59		100	52
66	100	57		100	53		100	48		100	45
67	100	56		100	54		100	60		100	57
68	100	51		100	64		100	54		100	61
69	100	43		100	35		100	46		100	39
70	100	57		100	59		100	55		100	63
71	100	57		100	49		100	62		100	52
72	100	58		100	51		100	66		100	58
73	100	62		100	59		100	72		100	69
74	100	53		100	52		100	58		100	67
75	100	65		100	76		100	51		100	56
76	100	58		100	54		100	62		100	59
77	100	55		100	56		100	56		100	68
78	100	54		100	52		100	58		100	56
79	100	61		100	56		100	59		100	61
80	100	62		100	56		100	66		100	68
81	100	59		100	70		100	66		100	61
82	100	58		100	60		100	60		100	63
83	100	53		100	54		100	65		100	62
84	100	58		100	66		100	57		100	57
85	100	60		100	53		100	58		100	55
86	100	66		100	59		100	64		100	60
87	100	59		100	57		100	55		100	58
88	100	63		100	59		100	65		100	57
89	100	61		100	60		100	65		100	58
90	100	61		100	51		100	52		100	57
91	100	63		100	66		100	59		100	54
92	100	56		100	59		100	52		100	58
93	100	61		100	57		100	64		100	59
94	100	52		100	53		100	60		100	68
95	100	55		100	56		100	64		100	61
96	100	55		100	61		100	60		100	59
97	100	64		100	77		100	40		100	44
98	100	56		100	56		100	52		100	57
99	100	60		100	55		100	55		100	58
100	100	56		100	54		100	66		100	56

Tabela 9 – McCormick- População 100 indivíduos

7.1.5 McCormick - População 200 cromossomos

	StandardAG					**TransgenicAG**			
seed	mut 3%	tempo		mut 5%	tempo	mut 3%	tempo	mut 5%	tempo
1	100	379		100	373	100	381	100	373
2	100	129		100	197	100	138	100	197
3	100	186		100	165	100	158	100	165
4	100	164		100	187	100	184	100	187
5	100	179		100	208	100	181	100	208
6	100	83		100	99	100	105	100	99
7	100	157		100	191	100	165	100	191
8	100	86		100	93	100	100	100	93
9	100	188		100	176	100	159	100	176
10	100	152		100	160	100	196	100	160
11	100	159		100	214	100	161	100	214
12	100	183		100	176	100	214	100	176
13	100	173		100	167	100	182	100	167
14	100	170		100	156	100	165	100	156
15	100	157		100	158	100	176	100	158
16	100	190		100	177	100	164	100	177
17	100	156		100	175	100	158	100	175
18	100	162		100	175	100	169	100	175
19	100	174		100	161	100	165	100	161
20	100	132		100	101	100	140	100	101
21	100	158		100	159	100	155	100	159
22	100	184		100	178	100	170	100	178
23	100	163		100	228	100	166	100	228
24	100	202		100	179	100	172	100	179
25	100	181		100	158	100	176	100	158
26	100	168		100	163	100	181	100	163
27	100	178		100	191	100	175	100	191
28	100	88		100	83	100	89	100	83
29	5	6		5	6	3	3	3	3
30	4	6		4	4	4	4	4	6
31	100	200		100	162	100	167	100	180
32	100	84		100	90	100	79	100	88
33	100	182		100	171	100	155	100	172
34	100	165		100	166	100	151	100	158

35	5	5		5	5		7	7		4	3
36	100	165		100	182		100	165		100	152
37	100	180		100	169		100	166		100	166
38	100	169		100	168		100	171		100	189
39	100	162		100	171		100	162		100	170
40	100	167		100	199		100	147		100	174
41	100	175		100	161		100	156		100	161
42	100	84		100	83		100	158		100	151
43	100	175		100	172		100	159		100	161
44	100	156		100	156		100	167		100	160
45	100	162		100	157		100	171		100	159
46	100	167		100	155		100	158		100	164
47	100	169		100	156		100	159		100	161
48	100	160		100	168		100	171		100	150
49	100	178		100	172		100	162		100	164
50	100	186		100	158		100	169		100	170
51	100	169		100	166		100	166		100	174
52	100	198		100	169		100	184		100	173
53	3	4		3	4		3	3		3	2
54	100	196		100	171		100	161		100	173
55	100	157		100	150		100	154		100	158
56	100	168		100	156		100	166		100	160
57	100	174		100	178		100	153		100	157
58	100	177		100	168		100	164		100	170
59	100	155		100	165		100	150		100	170
60	100	155		100	152		100	154		100	157
61	100	159		100	162		100	157		100	156
62	100	168		100	173		100	182		100	165
63	100	173		100	158		100	163		100	166
64	100	99		100	97		100	118		100	119
65	100	176		100	157		100	175		100	168
66	100	161		100	163		100	171		100	164
67	100	161		100	156		100	154		100	144
68	100	166		100	164		100	147		100	161
69	100	109		100	105		100	182		100	155
70	100	173		100	159		100	186		100	152
71	100	157		100	160		100	155		100	158

72	100	166		100	167		100	183		100	168
73	100	168		100	171		100	153		100	169
74	100	79		100	83		100	110		100	116
75	3	2		3	2		4	3		4	5
76	100	166		100	161		100	157		100	167
77	100	160		100	170		100	165		100	162
78	100	163		100	109		100	163		100	93
79	100	80		100	72		100	160		100	158
80	100	179		100	178		100	154		100	177
81	100	156		100	189		100	160		100	174
82	100	160		100	178		100	159		100	154
83	100	163		100	165		100	165		100	161
84	100	176		100	182		100	172		100	157
85	100	168		100	156		100	177		100	159
86	3	3		3	3		4	3		4	3
87	100	163		100	183		100	170		100	160
88	100	163		100	180		100	158		100	165
89	100	170		100	189		100	148		100	147
90	100	176		100	174		100	155		100	165
91	100	76		100	81		100	77		100	74
92	100	166		100	157		100	166		100	158
93	100	172		100	164		100	152		100	156
94	100	170		100	181		100	159		100	167
95	100	162		100	162		100	174		100	165
96	100	171		100	164		100	156		100	151
97	100	105		100	108		100	148		100	152
98	100	158		100	165		100	174		100	167
99	100	84		100	84		100	81		100	78
100	100	169		100	153		100	178		100	160

Tabela 10 – McCormick- População 200 indivíduos

7.1.6 McCormick - População 1000 cromossomos

	StandardAG					**TransgenicAG**				
seed	mut 3%	tempo		mut 5%	tempo	mut 3%	tempo		mut 5%	tempo
1	3	242		3	252	3	225		3	221
2	3	153		6	176	3	124		5	127

3	100	3152		100	2750		100	3340		100	2820
4	6	47		6	49		4	28		4	24
5	100	1275		100	1253		100	975		100	902
6	2	14		2	12		2	12		2	11
7	3	22		3	24		3	18		3	19
8	100	1003		100	1499		100	1032		100	1566
9	5	62		5	66		8	55		8	43
10	100	2837		100	2985		100	2653		100	2953
11	100	809		100	759		100	1645		100	773
12	5	60		5	76		4	23		4	24
13	4	39		4	52		3	21		3	16
14	100	949		100	1574		100	758		100	1448
15	3	18		3	33		5	34		5	25
16	100	1015		100	868		100	1146		100	861
17	5	45		5	60		7	89		7	83
18	5	53		5	45		7	71		7	67
19	3	16		3	17		4	28		4	24
20	100	826		100	827		100	949		100	909
21	6	41		6	34		6	37		6	34
22	100	2514		100	2461		100	2699		100	2725
23	100	2499		100	2693		100	2808		100	2529
24	100	1025		100	836		100	1668		100	830
25	5	37		5	35		4	23		4	20
26	7	88		7	94		8	66		8	64
27	100	2480		100	2548		100	3362		100	3208
28	4	26		4	28		3	16		3	18
29	4	26		4	26		4	24		4	23
30	3	20		3	21		3	16		3	16
31	7	97		7	95		100	2914		100	2832
32	5	32		5	39		3	16		3	10
33	4	29		4	31		5	32		5	24
34	100	861		100	899		100	2958		100	2846
35	7	84		5	86		8	77		5	70
36	100	2685		100	2508		100	2674		100	2571
37	100	2947		100	2630		100	2884		100	2592
38	3	21		3	19		3	18		3	16
39	3	19		3	18		3	17		3	12

40	3	21		3	24		3	16		3	15
41	4	29		4	25		3	17		3	16
42	2	14		2	10		2	9		2	8
43	5	34		5	33		5	20		5	22
44	100	841		100	793		100	2759		100	2725
45	100	751		100	767		100	2897		100	2722
46	100	2817		100	2649		100	2627		100	2509
47	5	46		5	39		5	29		5	20
48	4	26		4	26		3	16		3	16
49	6	44		6	42		5	38		5	32
50	100	2830		100	2597		100	2818		100	2537
51	3	23		3	25		4	17		4	16
52	6	40		6	52		7	40		7	43
53	3	27		3	22		3	26		3	18
54	100	2878		100	2749		100	2797		100	2517
55	100	2695		100	2726		100	3118		100	2718
56	5	36		5	43		3	35		3	26
57	100	976		100	1408		100	1207		100	1107
58	2	15		2	10		2	11		2	21
59	4	31		4	27		4	28		4	34
60	100	2839		100	2585		100	2789		100	2607
61	100	2873		100	2902		100	3143		100	3179
62	2	11		2	11		2	10		2	28
63	100	968		100	2501		100	1139		100	2100
64	2	16		2	13		2	10		2	31
65	3	20		3	18		4	17		4	19
66	5	36		5	36		8	62		8	55
67	100	1129		100	1244		100	879		100	938
68	5	41		5	39		6	37		6	32
69	100	1074		100	1184		100	1708		100	1981
70	2	10		2	10		2	9		2	24
71	6	46		6	42		6	36		6	35
72	1	5		1	4		1	3		1	10
73	6	43		6	42		6	41		6	44
74	100	2723		100	2682		100	2777		100	2990
75	3	21		3	20		5	32		5	88
76	100	2568		100	2548		100	2385		100	2463

77	4	35		4	27		4	25		4	33
78	100	2751		10	89		100	2624		10	73
79	100	2603		100	2644		100	2840		100	2728
80	5	50		5	49		6	45		6	34
81	100	3018		100	2500		100	920		100	2742
82	100	3143		100	1628		100	2779		100	2743
83	100	1216		100	2535		11	81		11	71
84	100	3555		100	2608		100	2691		100	2499
85	4	36		4	37		5	33		5	30
86	3	18		3	16		3	16		3	16
87	4	29		4	33		5	24		5	26
88	100	3249		100	2848		100	1144		100	2710
89	100	1435		100	2617		100	1486		100	1087
90	3	25		3	22		4	26		4	23
91	100	1953		100	1089		100	2776		100	1610
92	2	16		2	11		2	9		2	10
93	100	3230		100	1633		100	2629		100	2499
94	100	3321		100	2455		100	929		100	2704
95	5	40		5	36		7	45		7	47
96	100	2225		100	2653		100	2338		100	2702
97	4	52		4	50		4	30		4	27
98	100	2968		100	2746		100	2552		100	2550
99	3	20		3	18		3	17		3	16
100	5	38		5	32		5	34		5	33

Tabela 11 – McCormick- População 1000 indivíduos

7.1.7 Michalewicz - População 100 cromossomos

	StandardAG						**TransgenicAG**				
seed	mut 3%	tempo		mut 5%	tempo		mut 3%	tempo		mut 5%	tempo
1	100	139		100	130		100	167		100	158
2	8	18		5	16		4	14		100	59
3	100	52		100	55		100	69		100	72
4	6	5		6	4		100	41		100	45
5	100	45		100	47		100	51		100	54
6	1	3		1	2		1	2		1	1
7	2	3		2	2		2	1		2	1

8	100	41		100	43		100	46		100	52
9	100	56		100	50		100	48		100	62
10	6	3		6	3		5	3		5	4
11	100	39		100	43		100	41		100	44
12	100	34		100	31		100	40		100	38
13	100	36		100	34		100	37		100	41
14	7	7		7	5		6	4		6	3
15	100	32		100	35		100	33		100	38
16	100	31		100	43		100	32		100	31
17	5	3		5	4		4	2		4	2
18	100	33		100	37		100	42		100	49
19	100	37		100	35		100	34		100	36
20	6	6		6	7		100	44		100	53
21	6	2		6	2		100	45		100	42
22	8	6		8	5		4	2		4	1
23	100	40		100	42		100	32		100	34
24	100	31		100	39		100	36		100	33
25	100	37		100	35		100	32		100	31
26	100	41		100	41		100	32		100	35
27	100	33		100	41		100	35		100	35
28	4	1		4	1		5	1		5	1
29	6	1		6	1		5	2		5	1
30	4	3		4	3		6	3		6	4
31	4	1		4	1		5	2		5	2
32	5	2		5	2		4	1		4	2
33	4	1		4	1		100	31		100	40
34	4	1		4	1		3	1		3	1
35	7	4		5	3		6	2		5	2
36	7	1		7	1		8	1		8	1
37	100	37		100	33		100	39		100	40
38	100	33		100	37		100	54		100	33
39	100	34		100	43		100	32		100	42
40	5	1		5	1		4	1		4	1
41	100	42		100	44		100	35		100	32
42	100	36		100	37		100	38		100	30
43	100	30		100	35		100	32		100	35
44	100	44		100	34		100	38		100	29

45	100	32		100	36		100	35		100	30
46	100	36		100	37		100	41		100	35
47	5	2		5	3		7	3		7	3
48	100	31		100	37		100	37		100	34
49	100	41		100	36		100	31		100	34
50	5	1		5	1		100	31		100	33
51	100	37		100	41		100	31		100	32
52	4	1		4	1		5	1		5	1
53	100	58		100	56		100	69		100	65
54	100	57		100	55		100	34		100	31
55	100	35		100	38		100	42		100	51
56	5	1		5	1		4	1		4	1
57	100	35		100	39		100	45		100	30
58	7	1		7	2		3	2		3	1
59	5	1		5	2		4	2		4	1
60	100	39		100	38		100	51		100	31
61	100	42		100	36		100	39		100	28
62	5	1		5	1		3	1		3	1
63	5	1		5	1		5	1		5	2
64	100	25		100	26		100	30		100	22
65	100	35		100	28		100	32		100	31
66	5	1		5	1		4	1		4	1
67	5	1		5	2		5	1		5	1
68	4	1		4	1		4	1		4	1
69	4	2		4	1		5	1		5	1
70	100	34		100	40		100	35		100	30
71	100	36		100	35		100	34		100	31
72	6	4		6	5		7	5		7	6
73	100	33		100	34		100	42		100	48
74	100	40		100	35		100	31		100	29
75	100	32		100	34		100	30		100	34
76	6	2		6	2		5	1		5	1
77	100	40		100	35		100	34		100	43
78	4	1		100	38		4	1		100	23
79	100	37		100	34		100	31		100	32
80	5	1		5	1		4	1		4	1
81	100	32		100	36		100	30		100	36

82	5	1		5	1		5	1		5	1
83	100	37		100	35		100	30		100	33
84	4	1		4	1		3	1		3	1
85	5	1		5	1		5	1		5	2
86	100	33		100	37		100	39		100	32
87	100	40		100	34		100	32		100	45
88	5	1		5	1		5	1		5	1
89	100	35		100	36		100	40		100	35
90	100	41		100	35		100	40		100	33
91	4	3		4	4		4	3		4	6
92	100	32		100	34		100	30		100	23
93	4	1		4	2		5	1		5	2
94	100	35		100	34		100	38		100	38
95	6	3		6	2		7	3		7	4
96	100	34		100	35		100	46		100	43
97	100	24		100	27		100	24		100	19
98	5	1		5	1		5	1		5	1
99	100	32		100	36		100	39		100	41
100	100	34		100	38		100	35		100	41

Tabela 12 – Michalewicz - População 100 indivíduos

7.1.8 Michalewicz - População 200 cromossomos

	StandardAG					**TransgenicAG**				
seed	mut 3%	tempo		mut 5%	tempo		mut 3%	tempo		mut 5%	tempo
1	100	320		100	324		100	343		100	331
2	6	13		7	16		6	19		8	27
3	5	13		5	6		4	15		4	6
4	5	7		5	5		5	10		5	5
5	5	6		5	3		6	6		6	7
6	1	3		1	3		1	4		1	3
7	4	5		4	4		5	5		5	6
8	100	128		100	135		100	145		100	157
9	100	132		100	123		100	143		100	135
10	4	2		4	3		5	4		5	3
11	100	117		100	139		100	134		100	141
12	5	3		5	4		3	3		3	1

13	4	3		4	3		5	3		5	2
14	6	2		6	16		5	3		5	2
15	4	3		4	4		4	3		4	2
16	4	3		4	2		5	2		5	3
17	4	3		4	2		5	2		5	6
18	100	129		100	114		100	134		100	120
19	7	3		7	3		4	2		4	3
20	7	8		7	8		100	79		100	76
21	5	4		5	3		5	2		5	2
22	7	4		7	2		6	2		6	2
23	7	3		7	2		5	3		5	1
24	100	119		100	143		100	131		100	155
25	100	128		100	120		100	133		100	117
26	5	4		5	2		8	3		8	2
27	1	1		1	2		1	1		1	1
28	3	3		3	1		5	2		5	3
29	2	1		2	3		2	1		2	1
30	6	3		6	4		5	2		5	2
31	5	3		5	2		5	2		5	2
32	5	3		5	2		5	2		5	1
33	4	2		4	2		3	1		3	1
34	4	7		4	9		5	1		5	2
35	5	2		8	7		4	1		6	3
36	4	2		4	2		4	8		4	1
37	4	2		4	1		100	130		100	116
38	5	3		5	2		5	3		5	4
39	1	1		1	2		1	1		1	1
40	4	2		4	2		5	4		5	1
41	100	123		100	134		100	122		100	125
42	100	124		100	125		100	118		100	116
43	8	2		8	3		5	2		5	2
44	5	2		5	3		5	2		5	2
45	9	3		9	4		6	2		6	2
46	5	2		5	2		5	2		5	2
47	5	2		5	2		4	2		4	2
48	100	116		100	109		100	119		100	126
49	4	1		4	2		5	2		5	1

50	5	2		5	2		5	2		5	6
51	100	113		100	128		100	120		100	123
52	5	2		5	1		3	1		3	1
53	100	108		100	55		100	86		100	74
54	5	2		5	1		6	3		6	5
55	5	3		5	2		5	3		5	1
56	5	3		5	2		5	3		5	1
57	5	2		5	3		4	2		4	5
58	6	3		6	2		5	2		5	1
59	4	2		4	2		4	3		4	1
60	100	143		100	128		100	126		100	111
61	100	125		100	123		100	141		100	125
62	4	2		4	2		5	4		5	2
63	5	2		5	3		4	2		4	1
64	4	3		4	3		4	3		4	2
65	100	119		100	129		100	119		100	149
66	4	1		4	2		5	1		5	3
67	4	2		4	1		4	2		4	3
68	4	1		4	1		7	2		7	2
69	4	1		4	2		4	2		4	1
70	100	135		100	121		100	114		100	111
71	4	2		4	1		5	2		5	2
72	2	1		2	1		2	1		2	1
73	100	109		100	115		100	146		100	124
74	5	2		5	4		5	2		5	3
75	4	1		4	3		6	3		6	3
76	4	1		4	2		5	2		5	4
77	100	123		100	113		100	131		100	112
78	5	2		100	56		4	2		5	1
79	5	4		5	2		4	2		4	1
80	5	2		5	1		5	2		5	7
81	4	2		4	2		8	2		8	3
82	5	2		5	4		5	1		5	2
83	100	118		100	123		100	112		100	142
84	1	1		1	1		1	1		1	1
85	5	2		5	1		5	2		5	2
86	100	115		100	129		100	120		100	132

87	100	106		100	127		100	116		100	130
88	4	1		4	2		4	2		4	1
89	100	108		100	110		100	111		100	116
90	100	104		100	129		100	123		100	106
91	4	1		4	2		6	1		6	1
92	4	2		4	2		4	1		4	1
93	5	2		5	2		6	1		6	3
94	100	111		100	103		100	118		100	120
95	5	2		5	2		4	1		4	3
96	6	2		6	2		9	3		9	4
97	9	9		9	5		4	3		4	3
98	5	2		5	2		5	2		5	3
99	7	2		7	2		5	2		5	2
100	5	1		5	4		4	1		4	2

Tabela 13 – Michalewicz - População 200 indivíduos

7.1.9 Michalewicz - População 1000 cromossomos

	StandardAG						**TransgenicAG**				
seed	mut 3%	tempo		mut 5%	tempo		mut 3%	tempo		mut 5%	tempo
1	4	241		4	229		4	209		4	191
2	4	150		4	164		4	158		5	183
3	3	22		3	45		4	20		4	27
4	4	39		4	29		4	35		4	24
5	3	32		3	31		4	20		4	22
6	1	5		1	6		1	5		1	3
7	3	26		3	32		4	18		4	22
8	4	30		4	25		4	33		4	25
9	4	27		4	22		3	23		3	17
10	4	25		4	28		4	26		4	25
11	4	24		4	26		4	26		4	22
12	2	12		2	9		2	11		2	10
13	3	19		3	31		4	21		4	28
14	3	29		3	32		4	21		4	22
15	3	14		3	15		3	18		3	21
16	2	8		2	8		2	14		2	9
17	3	20		3	14		3	19		3	12

18	4	27		4	13		3	22		3	15
19	5	26		5	13		3	31		3	10
20	3	25		3	32		4	24		4	31
21	3	15		3	32		4	16		4	25
22	3	16		3	36		4	17		4	25
23	4	33		4	29		4	25		4	31
24	1	7		1	11		1	5		1	4
25	3	18		3	35		4	16		4	27
26	3	21		3	21		3	15		3	14
27	1	5		1	5		1	4		1	1
28	4	29		4	23		3	21		3	22
29	3	23		3	17		3	13		3	17
30	4	27		4	21		4	20		4	21
31	3	18		3	21		3	14		3	13
32	2	8		2	10		2	8		2	8
33	3	18		3	18		3	14		3	21
34	3	13		3	17		3	14		3	14
35	5	32		5	29		5	32		4	26
36	2	10		2	10		2	9		2	8
37	3	13		3	21		4	14		4	19
38	4	30		4	21		4	20		4	19
39	1	3		1	3		1	3		1	3
40	3	19		3	19		3	15		3	13
41	4	22		4	16		3	20		3	20
42	3	19		3	20		4	14		4	19
43	1	4		1	3		1	3		1	4
44	3	19		3	18		4	16		4	20
45	4	26		4	15		3	22		3	17
46	4	22		4	14		3	21		3	13
47	4	22		4	19		4	19		4	18
48	4	25		4	26		4	25		4	23
49	6	35		6	23		4	39		4	18
50	4	22		4	19		4	23		4	22
51	4	19		4	20		4	19		4	18
52	1	3		1	3		1	3		1	3
53	1	3		1	3		1	5		1	4
54	4	19		4	14		3	19		3	14

55	4	19		4	19		4	18		4	24
56	4	22		4	19		4	21		4	18
57	3	19		3	13		3	14		3	15
58	4	28		4	19		4	19		4	18
59	3	14		3	18		4	14		4	18
60	2	10		2	6		2	11		2	7
61	1	2		1	5		1	4		1	2
62	3	13		3	19		4	15		4	19
63	5	28		5	23		5	27		5	24
64	2	15		2	14		2	15		2	13
65	3	20		3	33		5	14		5	27
66	1	2		1	5		1	3		1	5
67	3	20		3	14		3	15		3	14
68	4	20		4	22		4	17		4	19
69	1	4		1	3		1	2		1	3
70	4	21		4	21		4	19		4	24
71	6	36		6	23		5	28		5	24
72	1	3		1	1		1	2		1	3
73	3	18		3	16		3	13		3	13
74	4	20		4	13		3	18		3	13
75	6	38		6	17		4	28		4	18
76	4	29		4	18		4	17		4	18
77	2	10		2	9		2	13		2	8
78	3	17		4	17		4	37		4	24
79	5	25		5	21		4	28		4	23
80	1	3		1	4		1	4		1	4
81	4	22		4	33		5	19		5	27
82	4	19		4	17		4	18		4	18
83	3	13		3	28		5	18		5	23
84	1	6		1	3		1	5		1	3
85	3	22		3	28		4	13		4	18
86	4	27		4	20		4	20		4	18
87	4	23		4	29		4	17		4	18
88	3	14		3	31		4	14		4	18
89	3	19		3	27		3	12		3	13
90	1	3		1	4		1	3		1	4
91	4	24		4	27		4	18		4	23

92	3	15		3	19		3	16		3	13
93	3	13		3	21		4	21		4	18
94	4	23		4	19		4	19		4	17
95	5	38		5	17		3	26		3	14
96	2	8		2	7		2	10		2	8
97	2	21		2	12		2	11		2	11
98	3	21		3	14		3	15		3	12
99	3	17		3	15		3	17		3	13
100	4	26		4	25		5	21		5	22

Tabela 14 – Michalewicz - População 1000 indivíduos