

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Andressa Oliveira Bernardes

**Algoritmo Genético aplicado ao jogo de
tabuleiro Sudoku**

Uberlândia, Brasil

2025

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Andressa Oliveira Bernardes

**Algoritmo Genético aplicado ao jogo de tabuleiro
Sudoku**

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção do título de Bacharel em Sistemas de Informação.

Orientadora: Prof^a. Dr^a. Christiane Regina Soares Brasil

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Sistemas de Informação

Uberlândia, Brasil

2025

Andressa Oliveira Bernardes

Algoritmo Genético aplicado ao jogo de tabuleiro Sudoku

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção do título de Bacharel em Sistemas de Informação.

Trabalho aprovado. Uberlândia, Brasil, 22 de setembro de 2025

Prof^a. Dr^a. Christiane Regina Soares
Brasil
Orientadora

Prof^a. Dr^a. Maria Camila Nardini Barioni

Prof. Dr. Paulo Henrique Ribeiro Gabriel

Uberlândia, Brasil
2025

Dedico este trabalho aos meus sobrinhos, Anaysa, Analya, Rayan e Rael.

Agradecimentos

Agradeço aos meus pais, Zilda e Wilson, por serem a base de tudo e por sempre me apoiarem e incentivarem nos estudos. Agradeço de coração à minha melhor amiga, Lara, por me encorajar, compreender e ser um ombro amigo desde a infância; ao meu namorado, Lucas, por me ajudar, escutar e trazer mais leveza e calma para a minha vida; e aos meus sobrinhos, Anaysa, Analya, Rayan e Rael, por serem a luz e a alegria dos meus dias. Amo todos vocês.

Um agradecimento especial ao meu professor de matemática do ensino médio, Turcato, que foi um apoio fundamental quando o cansaço batia e minhas contas só davam errado, e que certamente fez a diferença na minha formação.

Agradeço também à minha orientadora, Professora Christiane, cuja orientação, calma, paciência e didática foram fundamentais para a realização deste trabalho e para meu aprendizado ao longo das disciplinas que ministrou. Estendo meus agradecimentos a alguns professores marcantes da Universidade Federal de Uberlândia, em especial a Wendel e José Gustavo. Saibam que a dedicação de vocês faz a diferença na vida dos alunos. À Universidade Federal de Uberlândia, minha gratidão pela oportunidade e pelas condições de adquirir mais conhecimento.

Por fim, agradeço a Deus, por me dar forças todos os dias e me cercar de pessoas tão boas.

*“Às vezes, a vida é como esse túnel escuro: você não consegue ver a luz no fim,
mas, se continuar se movendo, chegará a um lugar melhor.”
(Avatar: a lenda de Aang)*

Resumo

Este trabalho aborda o desenvolvimento e a avaliação de um Algoritmo Genético (AG) para a resolução do quebra-cabeça Sudoku, um problema de otimização combinatória NP-completo. O objetivo principal foi investigar como diferentes parâmetros e operadores genéticos impactam a eficácia e a eficiência do algoritmo na busca por soluções válidas. Para isso, implementou-se um AG em Python com indivíduos representados por matrizes 9x9, função de *fitness* baseada na minimização de conflitos e operadores de recombinação, mutação e elitismo. A metodologia envolveu uma série de experimentos controlados para calibrar iterativamente os parâmetros do algoritmo. Os resultados definiram uma configuração otimizada com população de 600 indivíduos, mutação de 1%, elitismo de 5% e torneio de tamanho 3. A análise demonstrou que a escolha dos operadores foi o fator mais determinante para o desempenho, com a combinação da Recombinação Uniforme e da Mutação por Gene apresentando a abordagem mais eficaz para atingir a solução ótima com maior regularidade. Por fim, concluiu-se que a eficiência do AG é limitada pela sua natureza aleatória, tornando sua aplicação mais pertinente a cenários complexos onde etapas lógicas de pré-processamento não são, por si sós, suficientes para a resolução.

Palavras-chave: Algoritmo Genético, Sudoku, *Fitness*, Otimização de parâmetros.

Abstract

This study addresses the development and evaluation of a Genetic Algorithm (GA) for solving the Sudoku puzzle, an NP-complete combinatorial optimization problem. The main objective was to investigate how different parameters and genetic operators impact the effectiveness and efficiency of the algorithm in searching for valid solutions. To this end, a GA was implemented in Python with individuals represented by 9x9 matrices, a fitness function based on conflict minimization and recombination, mutation and elitism operators. The methodology involved a series of controlled experiments to iteratively calibrate the algorithm parameters. The results defined an optimized configuration with a population of 600 individuals, 1% mutation, 5% elitism and tournament size 3. The analysis demonstrated that the choice of operators was the most determining factor for performance, with the combination of Uniform Recombination and Gene Mutation presenting the most effective approach to achieving the optimal solution with greater regularity. Finally, it is concluded that the efficiency of GA is limited by its random nature, making its application more pertinent to complex scenarios where logical pre-processing steps are not, by themselves, sufficient for resolution.

Keywords: Genetic Algorithm, Sudoku, Fitness, Parameter optimization.

Lista de ilustrações

Figura 1 – Fluxograma de funcionamento de um AG.	17
Figura 2 – Exemplo de Sudoku clássico.	20
Figura 3 – Exemplo de Sudoku regular 4x4.	21
Figura 4 – Exemplo de Sudoku regular 6x6.	21
Figura 5 – Exemplo de Sudoku irregular.	22
Figura 6 – Exemplo de Sudoku diagonal.	22
Figura 7 – Exemplo de Killer Sudoku.	23
Figura 8 – Exemplo de <i>Dot</i> Sudoku.	24
Figura 9 – Exemplo de <i>Cross</i> Sudoku.	25
Figura 10 – Matriz base utilizada para testes.	53
Figura 11 – Matriz Sudoku Fácil 1.	55
Figura 12 – Matriz Sudoku Fácil 2.	55
Figura 13 – Matriz Sudoku Fácil 3.	56
Figura 14 – Matriz Sudoku Médio 1.	56
Figura 15 – Matriz Sudoku Médio 2.	57
Figura 16 – Matriz Sudoku Médio 3.	57
Figura 17 – Matriz Sudoku Difícil 1.	58
Figura 18 – Matriz Sudoku Difícil 2.	58
Figura 19 – Matriz Sudoku Difícil 3.	58

Lista de tabelas

Tabela 1 – Comparativo entre a abordagem de Neves e a desenvolvida neste trabalho.	33
Tabela 2 – Resultados para diferentes taxas de mutação.	35
Tabela 3 – Resultados para diferentes taxas de elitismo.	36
Tabela 4 – Resultados para diferentes taxas de torneio.	37
Tabela 5 – Resultados para o método de seleção por roleta.	38
Tabela 6 – Resultados para diferentes tamanhos de população.	39
Tabela 7 – Resultados comparativos entre funções de mutação e recombinação.	41
Tabela 8 – Comparação de resultados por matriz com pré-processamento.	44
Tabela 9 – Comparação de resultados por matriz sem pré-processamento.	47

Sumário

1	INTRODUÇÃO	13
1.1	Objetivo	13
1.2	Justificativa	14
1.3	Estrutura do texto	14
2	FUNDAMENTAÇÃO TEÓRICA	16
2.1	Algoritmos Evolutivos	16
2.1.1	Algoritmos Genéticos	16
2.1.1.1	Seleção	18
2.1.1.2	Recombinação	18
2.1.1.3	Mutação	18
2.1.1.4	Elitismo	19
2.2	Sudoku	19
2.2.1	Variações	21
2.3	Trabalhos Correlatos	25
2.3.1	Mantere e Koljonen (2007)	26
2.3.2	Deng e Li (2013)	26
2.3.3	Neves (2020)	26
2.3.4	Oliveira (2024)	27
2.4	Considerações Finais	27
3	DESENVOLVIMENTO	28
3.1	Arquitetura do Algoritmo Genético	28
3.1.1	Representação do indivíduo	28
3.1.2	Geração inicial	28
3.1.2.1	Geração inicial puramente aleatória	29
3.1.2.2	Geração inicial não-aleatória	29
3.1.3	Função de <i>fitness</i>	29
3.1.4	Método de seleção	30
3.1.4.1	Seleção por torneio	30
3.1.4.2	Seleção por roleta	30
3.1.5	Operador de recombinação	30
3.1.5.1	Recombinação uniforme	30
3.1.5.2	Recombinação de ponto único	31
3.1.6	Operador de mutação	31
3.1.6.1	Mutação uniforme	31

3.1.6.2	Mutação por troca	31
3.1.6.3	Mutação por gene	31
3.1.7	Elitismo	31
3.1.8	Ciclo de reprodução	32
3.1.9	Comparação da arquitetura dos AGs	32
3.2	Considerações Finais	33
4	RESULTADOS	34
4.1	Experimentos variando a taxa de mutação	34
4.1.1	Análise dos resultados	35
4.2	Experimentos variando a taxa de elitismo	36
4.2.1	Análise dos resultados	36
4.3	Experimentos com a seleção por torneio	37
4.3.1	Análise dos resultados	37
4.3.2	Experimentos com a seleção por roleta	38
4.3.3	Análise dos resultados	38
4.4	Experimentos variando o tamanho da população	38
4.4.1	Análise dos resultados	39
4.5	Experimentos com diferentes métodos de mutação e recombinação	40
4.5.1	Análise dos resultados	41
4.6	Experimentos comparando com a literatura	42
4.6.1	Comparação dos AGs com pré-processamento	42
4.6.1.1	Replicação da literatura	42
4.6.1.2	Modelo proposto	43
4.6.1.3	Análise dos resultados	44
4.6.2	Comparação dos AGs sem pré-processamento	46
4.6.2.1	Replicação da literatura	46
4.6.2.2	Modelo proposto	46
4.6.2.3	Análise dos resultados	47
4.7	Considerações Finais	48
5	CONCLUSÃO	49
	REFERÊNCIAS	50

APÊNDICES	52
APÊNDICE A – MATRIZ UTILIZADA COMO BASE NO DESENVOL- VIMENTO DO AG	53
ANEXOS	54
ANEXO A – MATRIZES UTILIZADAS NA COMPARAÇÃO COM A LITERATURA (SEÇÃO 4.6)	55

1 Introdução

Os Algoritmos Evolutivos (AEs) são técnicas computacionais inspiradas na Teoria da Evolução de [Darwin \(1859\)](#), utilizados em problemas complexos nos quais os métodos determinísticos enfrentam dificuldades em encontrar a solução ótima. O princípio fundamental dos AEs consiste em evoluir uma população de soluções candidatas por meio de um processo iterativo: indivíduos mais aptos são selecionados para gerar uma nova população por meio de operadores de seleção e/ou reprodução, como recombinação e mutação. Uma das classes mais notáveis de AEs são os Algoritmos Genéticos (AGs), que podem ser utilizados para otimização de diversos problemas combinatórios ([GOLDBERG, 1989](#)).

O quebra-cabeça lógico Sudoku é um exemplo canônico de problema combinatório de alta complexidade, que pode ter diversas variações (clássicos, diagonais, *killer*, entre outros) ([OLIVEIRA, 2024](#)). Ainda que as regras do Sudoku clássico sejam simples, ele é classificado como um problema NP (do inglês *Nondeterministic Polynomial time*)¹ ([YATO; SETA, 2003](#)), característica que demanda o uso de algoritmos capazes de explorar eficientemente em seu vasto espaço de soluções, o que torna os AGs uma abordagem promissora.

A literatura acadêmica indica que o sucesso dos AGs na resolução de Sudoku clássico depende principalmente da sua configuração. [Mantere e Koljonen \(2007\)](#) evidenciaram a viabilidade geral da abordagem, mostrando que AGs podem ser aplicados para resolver, avaliar e gerar os quebra-cabeças. Avançando na questão do desempenho, [Deng e Li \(2013\)](#) mostraram que a hibridização de operadores melhora a performance, embora com limitações. De forma complementar, a análise de [Neves \(2020\)](#) confirmou a viabilidade, mas apontou para desafios de eficiência computacional. Diante disto, o presente estudo dedicou-se a analisar o impacto de distintas parametrizações dos operadores genéticos sobre o desempenho do AG aplicado ao problema do Sudoku, buscando a otimização dos resultados.

1.1 Objetivo

O objetivo geral deste Trabalho de Conclusão de Curso foi desenvolver e otimizar um Algoritmo Genético para a resolução do jogo Sudoku clássico, investigando

¹Um problema está em NP se existe uma máquina de Turing não-determinística que pode resolvê-lo em tempo polinomial. Alternativamente, existe um algoritmo determinístico que pode verificar qualquer solução candidata válida em tempo polinomial ([GAREY; JOHNSON, 1979](#)).

como diferentes configurações de parâmetros e a escolha de operadores genéticos impactam seu desempenho.

Os objetivos específicos deste trabalho foram:

1. Desenvolver um Algoritmo Genético para o Sudoku, definindo a representação do problema e uma função de *fitness* para medir a qualidade de cada solução candidata.
2. Otimizar o desempenho do algoritmo por meio de experimentação controlada, calibrando seus parâmetros e comparando diferentes operadores genéticos para determinar a configuração mais eficaz.
3. Avaliar o modelo otimizado em um cenário com Sudokus de nível fácil, médio e difícil, comparando sua performance com uma abordagem da literatura para contextualizar seus resultados e sua relevância.
4. Comparar os resultados obtidos pelo Algoritmo Genético proposto neste trabalho com um trabalho de referência, com a utilização de um método de pré-processamento e sem a aplicação do mesmo, a fim de avaliar a eficiência dos dois modelos.

1.2 Justificativa

Este trabalho propôs a aplicação de uma heurística computacional – o Algoritmo Genético – para a resolução do problema do Sudoku, investigando sua eficiência na busca por soluções ótimas ou sub-ótimas. Embora seja conhecido como um passatempo ou jogo de lógica, a base do Sudoku possui várias aplicações no mundo real, tais como: escalonamento de funcionários ou tarefas sem sobreposição de recursos, planejamento de horários escolares (evitar conflitos entre turmas e professores), algoritmos de criptografia e verificação de integridade de dados, entre outros. Portanto, é um tema atual de suma importância podendo ser adaptado em diversos contextos, visando a otimização na resolução de questões complexas da atualidade.

1.3 Estrutura do texto

O presente trabalho está estruturado da seguinte forma:

- Capítulo 1: apresenta e contextualiza o problema, o objetivo e a justificativa deste Trabalho de Conclusão de Curso.

- Capítulo 2: apresenta os conceitos principais sobre Algoritmo Genético, o jogo Sudoku e os trabalhos relacionados ao tema.
- Capítulo 3: descreve a implementação do AG e as ferramentas utilizadas para seu desenvolvimento.
- Capítulo 4: discute e analisa os resultados obtidos a partir de experimentos realizados.
- Capítulo 5: relata as conclusões finais e sugestões de trabalhos futuros.

2 Fundamentação Teórica

Neste capítulo, serão apresentados os principais conceitos relacionados à elaboração deste Trabalho de Conclusão de Curso, como a definição de Algoritmos Evolutivos, a descrição do problema do quebra-cabeça Sudoku e uma coletânea de trabalhos referentes a este tema.

A seguir, será explicada a fundamentação teórica dos Algoritmos Evolutivos, focando nos Algoritmos Genéticos.

2.1 Algoritmos Evolutivos

Os Algoritmos Evolutivos (AEs) são técnicas de busca e otimização computacional fortemente inspiradas nos processos evolutivos que ocorrem na natureza, sendo baseados na teoria darwiniana da evolução (DARWIN, 1859). A partir de uma função a ser otimizada, denominada função de aptidão, gera-se aleatoriamente um conjunto de soluções candidatas (ou indivíduos) que serão avaliadas de acordo com sua qualidade, atribuindo-lhes um valor que mede a sua adequação: o *fitness* (GABRIEL; DELBEM, 2008).

Neste sentido, a evolução dos indivíduos ocorre de geração em geração, por meio dos operadores de seleção e de reprodução (recombinação e/ou mutação). A recombinação é aplicada, em geral, para duas soluções selecionadas (indivíduos pais) em que combinadas resultam em uma ou mais soluções (indivíduos filhos), enquanto que na mutação é efetuada uma perturbação, geralmente aleatória, sobre uma solução a fim de gerar outra solução nova. Estas novas soluções podem competir com os indivíduos da geração anterior, quando se é utilizado o elitismo. Este processo evolutivo é repetido iterativamente até que uma solução seja a ótima ou o número máximo de iterações seja obtido (GABRIEL; DELBEM, 2008).

A seguir, será descrito um tipo específico de Algoritmo Evolutivo: o Algoritmo Genético.

2.1.1 Algoritmos Genéticos

Dentre os diferentes tipos de AEs existentes, este trabalho utilizará o Algoritmo Genético (AG). Propostos por Holland (1975) e com suas primeiras aplicações bem-sucedidas demonstradas por Goldberg (1989), os AGs são hoje amplamente re-

conhecidos como uma classe de AEs. Essa classificação é sustentada por diversos autores, como [De Jong \(2006\)](#) e [Eiben e Smith \(2015\)](#), que os descrevem como "o mais conhecido tipo de algoritmo evolutivo".

A principal característica que define os AGs é a obrigatoriedade do uso do operador de recombinação (cruzamento ou *crossover*) para gerar descendentes, podendo combinar características das melhores soluções atuais. Enquanto outras estratégias evolutivas podem funcionar primariamente somente com o operador de mutação, o AG utiliza o cruzamento como sua força motriz para a exploração do espaço de busca, sendo a mutação um operador secundário, ainda que importante, para garantir a diversidade genética da população ([BACK, 1996](#)). Este processo iterativo, que evolui uma população de soluções ao longo de gerações, pode ser visualizado no fluxograma apresentado na Figura 1.

Figura 1 – Fluxograma de um AG genérico.



Fonte: Adaptado de [Corrêa et al. \(2010\)](#).

A grande relevância dos Algoritmos Genéticos reside em sua capacidade de

abordar problemas computacionalmente complexos, cujo vasto espaço de busca torna a exploração por métodos tradicionais inviável. Neste cenário, os AGs representam um grande avanço, com eficiência comprovada tanto em problemas clássicos de otimização combinatória — como o do caixeiro viajante (CUNHA, 2019) e o do percurso do cavalo (NASCIMENTO; BRASIL, 2024) — quanto em desafios logísticos modernos, como a otimização de manobras espaciais (DOMINGUES, 2024) e de malhas viárias (SCHMITZ, 2022). Sua aplicabilidade continua a se expandir para sistemas críticos, sendo adaptados para a tomada de decisão em desastres tecnológicos (GRANADOS; M.; NÚÑEZ, 2024), e para fronteiras inovadoras da ciência, como em abordagens híbridas com a computação quântica para identificação de danos (XU et al., 2025). Essa notável versatilidade os consolida como ferramentas poderosas para enfrentar a crescente complexidade dos problemas computacionais modernos.

A seguir, serão explicados os operadores genéticos utilizados pelos AGs, inspirados nos conceitos biológicos de reprodução.

2.1.1.1 Seleção

A seleção é o operador responsável por escolher quais indivíduos de uma população atual serão os progenitores da próxima geração. Este mecanismo funciona como uma força de "pressão seletiva", podendo favorecer os indivíduos com maior aptidão (ou seja, aqueles que representam melhores soluções para o problema), dependendo do tipo de seleção aplicada (DE JONG, 2006).

2.1.1.2 Recombinação

A recombinação é o principal operador genético de reprodução de um AG, que envolve dois genitores, em geral, para a criação de um novo descendente. Neste operador, seções dos cromossomos de cada pai são selecionadas e trocadas entre si, resultando em um genoma filho que é uma combinação das informações genéticas de seus progenitores (DE JONG, 2006).

2.1.1.3 Mutação

A mutação é um operador genético secundário de reprodução de um AG, que introduz diversidade na população. Este operador atua sobre um único indivíduo para criar um descendente. Portanto, o descendente é uma cópia deste indivíduo, com um ou mais de seus genes alterados aleatoriamente. A probabilidade de um gene sofrer essa alteração é determinada pela taxa de mutação (DE JONG, 2006).

2.1.1.4 Elitismo

O elitismo é uma estratégia complementar aplicada em AGs para garantir a preservação dos melhores indivíduos de uma geração para a seguinte. Nesta abordagem, um número de indivíduos de maior aptidão da população atual são copiados diretamente para a nova população, sem sofrer as alterações dos operadores de recombinação e mutação (EIBEN; SMITH, 2015). A quantidade de indivíduos a serem preservados é definida por um parâmetro, que pode ser um número fixo ou uma porcentagem da população total.

Na próxima seção, será descrito o jogo Sudoku, expondo suas regras e variantes.

2.2 Sudoku

O Sudoku foi baseado no jogo *Number Place* (GARNES, 1979), criado pelo arquiteto aposentado Howard Garns e publicado pela primeira vez em maio de 1979, no *Dell Pencil Puzzles and Word Games*. O jogo chegou ao Japão em 1984, onde foi batizado de Sudoku — que significa "números únicos", mas foi somente a partir do final de 2004 que sua publicação em diversos jornais consolidou sua popularidade mundial (DELAHAYE, 2006).

O Sudoku clássico é um quebra-cabeça matemático composto por uma matriz maior 9x9 e nove matrizes menores (dentro da maior) 3x3, como ilustrado na Figura 2. Nele, é dada uma quantidade de números fixos que varia dependendo da dificuldade do jogo. Quanto maior a dificuldade, menor é a quantidade de números que são pré-dispostos no início da partida.

Figura 2 – Sudoku Clássico.

7	3		5				2	8
8					9		6	
	9		3			4	1	5
3		1				6	7	
4		9		5		8		
	8			1			4	9
9		6	8		2	1	5	
	2		7	4	5			
5		3						4

Fonte: autora.

As regras do Sudoku são simples:

- Nenhum número pode se repetir em nenhuma coluna;
- Nenhum número pode se repetir em nenhuma linha;
- Nenhum número pode se repetir em nenhuma submatriz 3x3.

O jogo é vencido quando todos os números são encaixados em suas respectivas células.

A aplicação do Sudoku na computação vai além dos diversos algoritmos para sua geração, resolução e classificação, como os propostos por Oliveira (2024). O jogo já foi tema de um artigo da Microsoft sobre a aplicação de AEs (MCCAFFREY, 2016) e serve como um importante *benchmark* para vários problemas complexos do mundo moderno, sendo utilizado, por exemplo, para avaliar o desempenho e o consumo de energia em estudos de hardware neuromórfico (OSTRAU et al., 2022)².

Existem diversas variações do Sudoku, que serão mostradas na subseção a seguir.

²Hardware neuromórfico: sistemas de hardware especializados e projetados para acelerar a simulação de Redes Neurais de Disparo (*Spiking Neural Networks* - SNNs).

2.2.1 Variações

- Sudokus regulares: variação do jogo que possui uma matriz $N \times N$, sendo N um número inteiro positivo. O Sudoku clássico está nessa classificação (OLIVEIRA, 2024).

As Figuras 3 e 4 apresentam, respectivamente, exemplos de Sudokus regulares nos formatos 4x4 e 6x6.

Figura 3 – Sudoku regular 4x4.

			3
	4		
		3	2

Fonte: (SUDOKU-PUZZLES, 2025³).

Figura 4 – Sudoku regular 6x6.

	3		4		
		5	6		3
			1		
	1		3		5
	6	4		3	1
		1		4	6

Fonte: (SUDOKU-PUZZLES, 2025⁴).

- Sudokus irregulares: variação do Sudoku regular, que possui as mesmas regras básicas, em que as regiões possuem formatos não convencionais e podem estar desconectadas entre si (OLIVEIRA, 2024).

³<https://sudoku-puzzles.net/pt-br/sudoku-4x4-dificil/>

⁴<https://sudoku-puzzles.net/pt-br/sudoku-6x6-facil/>

A Figura 5 apresenta um exemplo de Sudoku irregular.

Figura 5 – Sudoku irregular.

		6	3		
	3			4	
5		3	1		4
2		4	6		1
	4			2	
		2	4		

Fonte: (PUZZLEPHIL, 2025⁵).

- Sudokus diagonais: variação do Sudoku regular que mantém as mesmas regras básicas, adicionando a restrição de que os números não podem se repetir nas diagonais principais da matriz maior (OLIVEIRA, 2024).

A Figura 6 apresenta um exemplo de Sudoku diagonal:

Figura 6 – Sudoku diagonal.

2	4	8	1	5	7	9	6	3
9	1	6	4	3	8	5	7	2
5	3	7	2	9	6	4	1	8
7	9	5	3	8	2	1	4	6
8	2	4	9	6	1	3	5	7
3	6	1	5	7	4	2	8	9
6	5	9	7	1	3	8	2	4
4	8	3	6	2	5	7	9	1
1	7	2	8	4	9	6	3	5

Fonte: autora.

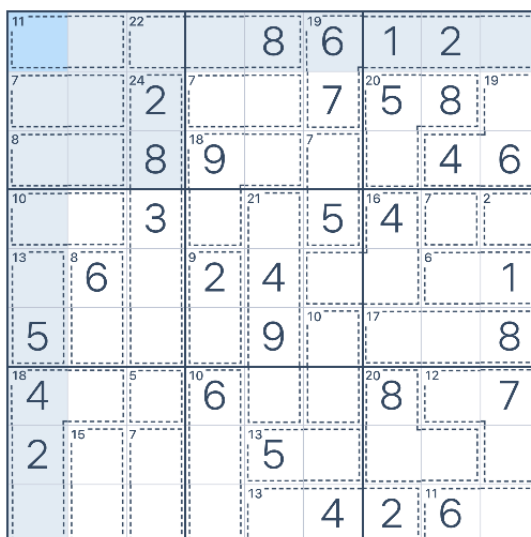
- Killer Sudoku: variação do Sudoku regular, que possui as mesmas regras básicas, em que é introduzido o conceito de “gaiola” – um grupo de células em uma

⁵<https://puzzlephil.com/puzzles/kikagaku/en/>

linha pontilhada cuja soma é mostrada como um pequeno número logo acima da gaiola. O objetivo é preencher os números de acordo com a soma de cada gaiola, sem que haja repetição dentro delas

A Figura 7 apresenta um exemplo de Killer Sudoku:

Figura 7 – Killer Sudoku.



Fonte: (SUDOKU, 2025⁶).

- Outras variações: o Sudoku regular possui diversas variações com características específicas. Algumas variações possuem layouts diferenciados, como formatos que remetem a figuras (borboletas e flores), matrizes que se intersectam ou até mesmo os centros que formam uma nova submatriz.

Nas Figuras 8 e 9 estão alguns exemplos de Sudokus diversos:

⁶<<https://sudoku.com/br/killer>>

Figura 8 – Dot Sudoku.

	1			2				8
	5	2			3		1	9
8	6	9	5	1			3	
6		3	7	5	8	9	2	1
		7	2			5	4	
2	9	5	6		1	3		
9	3	6	1	8			7	
4	2			7	6	8		
5	7	8	4	9		1	6	3

Fonte: (SUDOKU-PUZZLES, 2025⁷).

⁷<<https://sudoku-puzzles.net/pt-br/center-dot-sudoku-facil/>>

Figura 9 – Cross Sudoku.

						3	8	5			6	4	2	7					
						9	6			8	2	7			5	3			
						2	7	4		9	5	3			6				
						5	9			2	3	8			1	4			
							2						9	6			5		
						7		8		6	1			9	3	2			
	8	5							9	3	8	6	5				3		4
9		1	2	8	5		3	7	5	9	2	1	4				6	2	
3	7	6	4	9	1	8	5	2	4	7	1	3	9	6			7		
	6	2	8	1		5		4			7	6		3	2	1	7	9	4
1	9	8		4	7	3	2				4	9	7	1	4	8	5	2	3
5	3	4		6	2	7	8	1		6	3	4	5	2		3	6		8
4	1			2		9	6	5	7	3		2	1			5	3	6	9
6	5	3	1		9		4	8	6	1	9	7	3		6	2	9		1
8		9		5	4	1	7	3	2	4	5		6	9	7	4		5	
									3	8		5	7						
						3	5	2	4	9	7		8						
						7				2	1		9	3					
							3	4				9							
									9		4								
						6	2		8			1		7					

Fonte: (SUDOKU-PUZZLES, 2025⁸).

Neste Trabalho de Conclusão de Curso foi utilizado o Sudoku clássico.

Na próxima seção, serão apresentados alguns trabalhos relacionados ao problema do jogo Sudoku focando na aplicação de AGs.

2.3 Trabalhos Correlatos

Esta seção apresenta os trabalhos relacionados à resolução do Sudoku, cuja análise foi fundamental para a elaboração deste Trabalho de Conclusão de Curso.

⁸<<https://sudoku-puzzles.net/pt-br/cross-sudoku-facil/>>

2.3.1 Mantere e Koljonen (2007)

O objetivo do artigo "*Solving, Rating and Generating Sudoku Puzzles with GA*" (MANTERE; KOLJONEN, 2007) foi investigar a aplicação de AGs na resolução, avaliação e geração de quebra-cabeças Sudoku. Foi utilizado um AG para resolver Sudokus existentes, avaliar a dificuldade deles e gerar novos quebra-cabeças, verificando se eles têm uma solução única. Os autores testaram se a dificuldade atribuída aos quebra-cabeças Sudoku em jornais correspondia à dificuldade observada na otimização por AGs e os resultados confirmaram essa correspondência, se mostrando eficientes em resolvê-los e gerar novos Sudokus. Esse estudo reforça a viabilidade de AGs para resolução de problemas combinatórios, especialmente no contexto do quebra-cabeça Sudoku.

2.3.2 Deng e Li (2013)

O objetivo do artigo "*A novel hybrid genetic algorithm for solving Sudoku puzzles*" (DENG; LI, 2013) foi melhorar o desempenho de um AG através de uma hibridização para resolução do quebra-cabeça Sudoku. Nele, os operadores de seleção, *crossover* e mutação são aprimorados, considerando as características do jogo. Os resultados das simulações demonstraram que o AG híbrido apresentou melhorias significativas na taxa de convergência e estabilidade. Eles foram satisfatórios para Sudokus de baixa e média dificuldade, entretanto, para Sudokus difíceis e extremamente difíceis, o desempenho do algoritmo foi limitado. Esse estudo utiliza uma abordagem semelhante à que se pretende utilizar no presente projeto e para o mesmo propósito, sendo útil como base teórica relevante.

2.3.3 Neves (2020)

O objetivo do trabalho de conclusão de curso "Análise da eficiência de um Algoritmo Genético aplicado ao Sudoku" (NEVES, 2020) foi avaliar o desempenho de um AG na resolução do quebra-cabeça Sudoku. Nele, é desenvolvido e analisado um AG quanto ao seu funcionamento e eficiência. Os resultados de Neves (2020) mostram que a abordagem é uma alternativa viável, ainda que passível de melhorias. Este estudo relaciona-se diretamente ao presente projeto, pois aborda o mesmo problema (Sudoku) com a mesma técnica (AG), fornece uma metodologia detalhada para a implementação e para análise da eficiência do algoritmo e investiga como a variação dos parâmetros impacta diretamente o desempenho.

2.3.4 Oliveira (2024)

O objetivo da monografia "Sudoku: Um estudo sobre algoritmos de geração, classificação e resolução para aplicação educacional" (OLIVEIRA, 2024) foi explorar o uso do Sudoku no ensino de pensamento computacional, um componente curricular da Base Nacional Comum Curricular, visto que é um jogo de raciocínio lógico com diversas variações. Nele, são propostos e implementados três algoritmos: um para geração, outro para classificação e um terceiro para resolução, utilizado para validação dos algoritmos anteriores. Os resultados demonstraram que o algoritmo proposto é eficiente para resolver Sudokus de ordens 4, 6 e 9, sendo especialmente eficaz para quebra-cabeças mais simples. A classificação de Sudokus também foi satisfatória, porém houve limitação na geração de Sudokus únicos. Esse estudo também trata a aplicação de algoritmos para resolução do quebra-cabeça Sudoku e, apesar de AGs não serem utilizados nele, ele será utilizado como base teórica.

O trabalho intitulado "Análise da eficiência de um algoritmo genético aplicado ao Sudoku" (NEVES, 2020) é a referência central desta pesquisa, pois oferece uma análise completa da aplicação desta técnica para resolver o problema proposto. Sua metodologia detalha a implementação das funções essenciais do algoritmo, como *Fitness*, Seleção, *Crossover* e Mutação. A principal contribuição do estudo reside na rigorosa análise de eficiência, que demonstra como a variação de parâmetros impacta o desempenho do algoritmo. Assim, a obra não apenas fornece resultados concretos para comparação, mas também evidencia a calibração dos operadores como um fator crítico para a obtenção de uma solução eficiente.

2.4 Considerações Finais

Este capítulo apresentou os fundamentos dos AGs e sua vasta aplicação em problemas complexos. Em seguida, o Sudoku foi caracterizado como um problema de otimização combinatória, cujo espaço de busca desafiador o torna um objeto de estudo ideal. Além disto, foram apresentados diversos tipos de Sudoku, destacando que neste Trabalho de Conclusão de Curso foi utilizado o Sudoku clássico. A análise de trabalhos correlatos reforçou a viabilidade do uso de AGs para a resolução do Sudoku, ao mesmo tempo que evidenciou a importância das escolhas de modelagem desses algoritmos.

3 Desenvolvimento

Neste Trabalho de Conclusão de Curso, o desenvolvimento do Algoritmo Genético para resolução do quebra-cabeça Sudoku clássico foi realizado na linguagem de programação Python (versão 3.12.3), devido à sua versatilidade, facilidade de implementação e disponibilidade de bibliotecas. O ambiente de desenvolvimento utilizado foi o *Visual Studio Code*, juntamente com as extensões *Python*, *Pylance* e *Python Debugger* e as bibliotecas *Struct*, *Time*, *Numba* e *NumPy*. O sistema operacional utilizado foi o Linux Mint.

O código-fonte completo está disponível publicamente no GitHub, com acesso por meio do seguinte link: <https://github.com/andressaob/UFU/tree/main/semestre8/TCC>.

3.1 Arquitetura do Algoritmo Genético

A seguir, apresenta-se a estrutura do AG modelado para solucionar o quebra-cabeça Sudoku clássico, com a descrição dos métodos e operadores genéticos que compõem a solução.

3.1.1 Representação do indivíduo

Cada indivíduo representa uma solução candidata para o problema do Sudoku, encapsulado em uma classe, que é composta por dois atributos: uma matriz 9x9 contendo as suas configurações e um valor de *fitness* que qualifica esta solução.

A matriz é implementada utilizando a biblioteca *NumPy*, o que otimiza as operações de manipulação e cálculo. As células da matriz que correspondem aos números fixos do problema inicial permanecem inalteradas durante todo o processo evolutivo. As demais células, que iniciam vazias, são preenchidas e modificadas pelos operadores genéticos a cada geração, constituindo os genes que evoluem. Estas células são representadas na matriz inicial com zeros. A matriz inicial utilizada como base está no Apêndice A.

3.1.2 Geração inicial

A população inicial compreende o primeiro conjunto de indivíduos (matriz de vetores) do AG. Neste trabalho, tem-se duas possibilidades para geração da população inicial: puramente aleatória e não-aleatória.

3.1.2.1 Geração inicial puramente aleatória

Para cada indivíduo a ser criado na população, o processo consiste em copiar a matriz base do problema, com suas células fixas, e preencher todas as células restantes (não fixas) com números inteiros sorteados aleatoriamente no intervalo de 1 a 9.

É importante ressaltar que, devido a essa inicialização puramente aleatória dos números, a grande maioria dos indivíduos da primeira geração não satisfaz as regras fundamentais do Sudoku. A convergência para soluções válidas e otimizadas é responsabilidade do processo evolutivo subsequente, guiado pela função de *fitness* e pelos operadores genéticos.

3.1.2.2 Geração inicial não-aleatória

Para a etapa de comparação com a literatura, implementou-se uma função de geração inicial não-aleatória. Neste método, os indivíduos são criados sem a repetição de números em suas submatrizes 3x3, satisfazendo desde o início uma das restrições fundamentais do Sudoku. Esta abordagem foi aplicada exclusivamente nos experimentos cujos resultados são descritos na Seção 4.6.

3.1.3 Função de *fitness*

A função de *fitness*, ou função de aptidão, é o componente do algoritmo responsável por avaliar a qualidade de cada indivíduo em relação ao objetivo do problema. O objetivo é medir o quão próximo um indivíduo está de ser uma solução válida para o Sudoku.

O cálculo do *fitness* é definido como a soma total de violações das regras do jogo. A função opera da seguinte forma:

1. Uma variável contadora de conflitos é inicializada com zero.
2. O algoritmo itera sobre cada uma das 9 linhas da matriz, contando a quantidade de números duplicados. Para cada número que se repete, o contador de conflitos é incrementado.
3. O mesmo processo é repetido para cada uma das 9 colunas.
4. Por fim, a verificação é realizada para cada uma das 9 submatrizes 3x3.

O valor final retornado pela função é o número total de conflitos. Portanto, um indivíduo com *fitness* igual a 0 representa uma solução ótima, que não viola nenhuma regra do Sudoku. O objetivo do AG é minimizar esse valor até que ele chegue a zero.

3.1.4 Método de seleção

O método de seleção determina quais indivíduos da população atual serão escolhidos para se tornarem pais da próxima geração. A seguir, serão detalhados os métodos de seleção que foram utilizados no presente trabalho.

3.1.4.1 Seleção por torneio

Neste método, a seleção de um pai ocorre através de um "torneio". Para cada seleção, um grupo de k indivíduos é sorteado aleatoriamente da população, e o competidor com o melhor *fitness* dentro desse grupo é declarado o vencedor. O indivíduo vencedor é então escolhido para a reprodução, e o processo se repete até que o número necessário de pais seja formado.

3.1.4.2 Seleção por roleta

Este método simula uma roleta onde cada indivíduo possui uma fatia com tamanho proporcional à sua aptidão (*fitness*). Indivíduos mais aptos recebem fatias maiores, o que aumenta sua probabilidade de serem selecionados a cada "giro". Embora favoreça as melhores soluções, a roleta ainda garante uma chance de seleção aos indivíduos menos aptos, o que ajuda a manter a diversidade da população.

3.1.5 Operador de recombinação

A recombinação, ou *crossover*, é o processo pelo qual dois indivíduos pais geram um ou mais filhos, combinando suas características genéticas. A seguir, serão descritos os operadores de recombinação utilizados neste trabalho.

3.1.5.1 Recombinação uniforme

Neste método, o cromossomo do filho é construído gene a gene. Para cada posição, o algoritmo realiza uma escolha aleatória, com 50% de chance para cada pai, para definir de qual genitor o gene correspondente será herdado. Esse processo resulta em um filho que representa um mosaico das características genéticas de seus pais.

3.1.5.2 Recombinação de ponto único

Este método seleciona um ponto de corte aleatório, que divide o cromossomo de cada pai em dois segmentos. O filho é então gerado pela junção do primeiro segmento do primeiro pai com o segundo segmento do segundo pai, criando um novo indivíduo que combina grandes blocos de genes de ambos os genitores.

3.1.6 Operador de mutação

A mutação é um operador genético que introduz uma diversidade adicional na população, gerada por meio de uma perturbação sobre um ou mais indivíduos. A seguir, serão descritos os operadores de mutação aplicados neste trabalho.

3.1.6.1 Mutação uniforme

Neste método, cada indivíduo da população passa por um teste probabilístico, baseado na taxa de mutação. Caso um indivíduo seja selecionado, um de seus genes não fixos é sorteado aleatoriamente e seu valor é substituído por um novo número inteiro, de 1 a 9.

3.1.6.2 Mutação por troca

De forma semelhante à mutação uniforme, cada indivíduo tem uma chance de ser selecionado para mutação. Se isso ocorrer, o algoritmo sorteia dois genes (posições) do indivíduo e troca seus valores entre si.

3.1.6.3 Mutação por gene

Nesta mutação, o teste probabilístico é aplicado a cada gene individualmente em toda a população. Se um gene específico for selecionado para mutar, seu valor é substituído por outro, sorteado aleatoriamente no mesmo intervalo, de 1 a 9.

3.1.7 Elitismo

O elitismo é uma estratégia utilizada para garantir que as melhores soluções encontradas em uma geração não sejam perdidas na geração seguinte.

No início de cada ciclo reprodutivo, esta função primeiro ordena a população inteira com base no *fitness*, do melhor ao pior. Em seguida, ela calcula um número absoluto de indivíduos correspondente à taxa de elitismo definida (ex: 10% de uma

população de 600 indivíduos resulta em uma elite de 60). Estes indivíduos da elite, os mais aptos da geração, são copiados diretamente para a nova população, sem passar pelos processos de seleção, recombinação ou mutação.

3.1.8 Ciclo de reprodução

O ciclo de reprodução é o motor que impulsiona a evolução a cada geração. Ele executa uma sequência ordenada de passos para criar uma nova população a partir da atual:

1. Aplicação do elitismo: a função elitismo é chamada para ordenar a população atual e determinar a quantidade de indivíduos da elite. Esses indivíduos são transferidos diretamente para a nova população.
2. Geração de novos indivíduos: em seguida, um laço de repetição preenche o restante das vagas na nova população. Em cada iteração deste laço, dois pais são selecionados da iteração atual através do método de seleção, que passam pelo operador de recombinação para gerar um novo filho e esse filho pode ser submetido ao operador de mutação. O *fitness* do filho é calculado e ele é inserido na nova população.
3. Atualização da população: ao final do processo, a população antiga é completamente substituída pela nova população, e o ciclo está pronto para recomeçar na próxima geração até que encontre a solução ótima ou o número máximo de gerações seja atingido.

Na próxima subseção, será apresentada uma tabela com a síntese do AG desenvolvido e do trabalho de referência (NEVES, 2020), a fim de facilitar a comparação entre os dois modelos.

3.1.9 Comparação da arquitetura dos AGs

As estruturas e os métodos utilizados no presente trabalho e no de Neves (2020) foram sintetizados na Tabela 1, que apresenta um comparativo direto das abordagens.

Tabela 1 – Comparativo entre a abordagem de Neves e a desenvolvida neste trabalho.

CRITÉRIO	Neves (2020)	Este trabalho
GERAÇÃO INICIAL	Matriz numérica com preenchimento aleatório nas submatrizes sem repetição.	Matriz numérica com preenchimento aleatório.
MÉTODO DE SELEÇÃO	Seleção por roleta.	Seleção por torneio.
MÉTODO DE RECOMBINAÇÃO	Ponto Único.	Uniforme.
MÉTODO DE MUTAÇÃO	Mutação por Troca dentro das submatrizes.	Mutação por Gene.
FUNÇÃO DE FITNESS	Soma de valores duplicados por linha, coluna e submatriz.	Soma de valores duplicados por linha, coluna e submatriz.

Fonte: A autora.

3.2 Considerações Finais

Este capítulo detalhou a implementação do Algoritmo Genético para a resolução do Sudoku clássico, abordando desde a configuração do ambiente até a arquitetura da solução. Foram definidos seus componentes essenciais: a representação do indivíduo e a função de *fitness* baseada na minimização de conflitos. Para permitir uma análise experimental robusta, foram implementados e detalhados múltiplos métodos para cada operador genético — incluindo diferentes abordagens para seleção, recombinação e mutação — complementados por uma estratégia de elitismo. Por fim, apresentou-se uma tabela comparativa entre o modelo proposto e a abordagem utilizada como referência principal. Com o algoritmo totalmente funcional e parametrizável, estabelece-se a base metodológica para a próxima etapa, que consiste na execução dos experimentos e na análise comparativa dos resultados, a ser apresentada no capítulo seguinte.

4 Resultados

Este capítulo abordará a descrição dos experimentos realizados e a análise dos resultados obtidos neste Trabalho de Conclusão de Curso. Uma série de experimentos controlados foi conduzida para determinar a configuração mais indicada para o AG aplicado ao problema do quebra-cabeça Sudoku clássico.

Os testes foram realizados em um notebook com 8GB de memória RAM, 512GB de armazenamento interno, placa de vídeo Intel Corporation Device 4628 e processador 12th Gen Intel Core i5-1235U × 10. O sistema operacional utilizado foi o Linux Mint.

Para garantir a comparabilidade entre os testes e estabelecer uma metodologia consistente, os seguintes parâmetros foram utilizados nos experimentos:

- Número de execuções: 10 para cada configuração, visando garantir a consistência estatística dos resultados.
- Critério de parada: número de gerações igual a 200 ou quando o AG encontra a solução ótima.
- Instância: foi utilizada uma matriz de nível fácil, denominada “Matriz Base”, elaborada para os testes com as taxas dos parâmetros do AG, que pode ser encontrada no Apêndice [A](#).

A seguir, as próximas seções detalham os experimentos realizados.

4.1 Experimentos variando a taxa de mutação

O primeiro experimento teve como objetivo determinar a taxa de mutação mais adequada para este AG, buscando um equilíbrio entre a exploração de novas soluções e a convergência do algoritmo. Para este teste, os seguintes parâmetros foram mantidos fixos:

- Tamanho da população: 600 indivíduos,
- Taxa de elitismo: 5%,
- Seleção por torneio, com tamanho igual a 3.

A taxa de mutação foi o parâmetro a ser testado, variando com os valores de 0,5%, 1%, 3% e 5%. Os dados consolidados deste experimento estão apresentados na Tabela 2, onde E = elitismo e T = torneio.

Tabela 2 – Resultados para diferentes taxas de mutação.

E = 5 e T = 3	Mutação = 0,5%	Mutação = 1%	Mutação = 3%	Mutação = 5%
População	Melhor <i>fitness</i>	Melhor <i>fitness</i>	Melhor <i>fitness</i>	Melhor <i>fitness</i>
600	3	4	5	6
	6	7	2	7
	2	4	4	2
	7	8	8	9
	8	6	6	2
	7	4	4	5
	5	2	2	8
	4	2	8	8
	2	4	5	4
	8	4	9	6
Média	5,2	4,5	5,3	5,7
Desvio Padrão	2,35	1,96	2,45	2,45
Melhor	2	2	2	2
Pior	8	8	9	9

Fonte: autora.

4.1.1 Análise dos resultados

A análise dos dados da Tabela 2 revela que a taxa de mutação de 1% apresentou o melhor desempenho, pois alcançou a menor média de melhor *fitness* (4,5), indicando que, em média, as soluções geradas com esta taxa foram as mais próximas da solução ótima (*fitness* igual a 0).

A taxa de 1% também foi a que apresentou o menor desvio padrão (1,96) entre os resultados de *fitness*, que é um indicador crucial de confiabilidade. As outras taxas não apenas resultaram em uma média de *fitness* maior, mas também um desvio padrão mais alto, indicando maior variabilidade nos resultados.

Com base nesta análise, embora nenhuma das configurações tenha convergido para a solução ótima, a taxa de mutação de 1% foi selecionada para ser o valor padrão para os testes subsequentes.

4.2 Experimentos variando a taxa de elitismo

Uma vez definida a melhor taxa de mutação como 1% para este problema (conforme os resultados da Tabela 2), o foco do segundo experimento foi ajustar a taxa de elitismo. O propósito deste teste foi identificar o percentual ideal de indivíduos de elite a serem preservados entre as gerações.

Mantendo os demais parâmetros da configuração anterior (torneio = 3 e população de 600 indivíduos), a taxa de elitismo foi o parâmetro variável, testada com os valores de 5%, 10%, 15% e 20%. A Tabela 3 sintetiza os resultados obtidos, onde M = mutação e T = torneio.

Tabela 3 – Resultados para diferentes taxas de elitismo.

M = 1 e T = 3	Elitismo = 5%	Elitismo = 10%	Elitismo = 15%	Elitismo = 20%
População	Melhor <i>fitness</i>	Melhor <i>fitness</i>	Melhor <i>fitness</i>	Melhor <i>fitness</i>
600	4	2	4	4
	7	4	8	7
	4	4	2	6
	8	7	6	2
	6	5	6	2
	4	9	10	7
	2	10	2	2
	2	3	4	5
	4	4	5	8
	4	13	6	5
Média	4,5	6,1	5,3	4,8
Desvio Padrão	1,96	3,54	2,50	2,25
Melhor	2	2	2	2
Pior	8	13	10	8

Fonte: A autora.

4.2.1 Análise dos resultados

Na análise da taxa de elitismo, exposta na Tabela 3, observou-se que a taxa de 5% obteve a menor média de melhor *fitness* (4,5), assim como o menor desvio padrão (1,96), demonstrando ser a configuração mais estável e com o melhor desempenho.

Assim como nos resultados anteriores, a convergência para a solução ótima não foi atingida. Entretanto, a taxa de elitismo de 5% destacou-se por sua performance superior e consistência, sendo, portanto, definida como o parâmetro padrão para a continuidade do estudo.

4.3 Experimentos com a seleção por torneio

Após a definição da taxa de mutação (1%) e da taxa de elitismo (5%) nos experimentos anteriores, o terceiro experimento teve como foco analisar a dimensão do torneio. O objetivo foi determinar o tamanho de torneio mais adequado para selecionar o indivíduo mais apto (menor *fitness*) da população e comparar o desempenho deste método com a seleção por roleta.

Para o experimento, o tamanho do torneio foi o parâmetro variável, testado com os valores 3, 4 e 5. Os resultados foram compilados na Tabela 4.

Tabela 4 – Resultados para diferentes taxas de torneio.

M = 1 e E = 5	Torneio = 3	Torneio = 4	Torneio = 5
População	Melhor <i>fitness</i>	Melhor <i>fitness</i>	Melhor <i>fitness</i>
600	4	6	6
	7	7	8
	4	5	4
	8	6	6
	6	6	6
	4	4	4
	2	10	6
	2	2	9
	4	8	9
	4	4	9
Média	4,5	5,8	6,7
Desvio Padrão	1,96	2,25	1,95
Melhor	2	2	4
Pior	8	10	9

Fonte: A autora.

4.3.1 Análise dos resultados

Conforme a análise da Tabela 4, o tamanho de torneio 3 destacou-se como a configuração mais eficaz. Ele não apenas obteve a menor média de melhor *fitness* (4,5), mas também apresentou um dos menores desvios padrão (1,96), indicando alta consistência.

Embora, novamente, a solução ótima não tenha sido alcançada, o tamanho de torneio 3 foi escolhido como referência para os próximos testes devido ao seu desempenho superior e estabilidade.

4.3.2 Experimentos com a seleção por roleta

Neste experimento, realizou-se a comparação entre a seleção por torneio e o método de seleção por roleta. Para tal, o método da roleta foi testado, considerando a taxa de mutação de 1% e elitismo de 5%. Os resultados obtidos estão presentes na Tabela 5.

Tabela 5 – Resultados para o método de seleção por roleta.

M = 1 e E = 5	Roleta
População	Melhor <i>fitness</i>
600	10
	9
	2
	4
	6
	5
	6
	4
	11
	7
Média	6,4
Desvio Padrão	2,88
Melhor	2
Pior	11

Fonte: A autora.

4.3.3 Análise dos resultados

A análise dos dados confirmou a superioridade da seleção por torneio sobre o método da roleta. O torneio apresentou tanto um *fitness* médio menor (4,5 contra 6,4) quanto um desvio padrão mais baixo (1,96 contra 2,88), indicando resultados melhores e mais consistentes. Por essa razão, a seleção por torneio foi mantida como o método padrão para os testes seguintes.

4.4 Experimentos variando o tamanho da população

A etapa final de experimentos focou no tamanho da população, um parâmetro que lida com o equilíbrio entre a exploração do espaço de busca e a velocidade de convergência. Este teste foi realizado a fim de analisar se poderia ser encontrado um

resultado melhor com uma população menor que 600, uma vez que quanto menor a população, menos gasto computacional.

Com os parâmetros de mutação (1%), elitismo (5%) e torneio (3) já definidos nos testes anteriores, o parâmetro variável foi o tamanho da população. Para isto, os tamanhos de população escolhidos para serem testados foram 200, 400 e 600. A seguir, são apresentados na Tabela 6 os resultados obtidos.

Tabela 6 – Resultados para diferentes tamanhos de população.

M=1, E=5, T=3	População = 200	População = 400	População = 600
Execução	Melhor <i>fitness</i>	Melhor <i>fitness</i>	Melhor <i>fitness</i>
1	15	7	4
2	16	5	7
3	11	8	4
4	12	6	8
5	18	9	6
6	12	8	4
7	8	9	2
8	8	10	2
9	9	9	4
10	12	7	4
Média	12,1	7,8	4,5
Desvio Padrão	3,38	1,55	1,96
Melhor	8	5	2
Pior	18	10	8

Fonte: A autora.

4.4.1 Análise dos resultados

A análise da Tabela 6 aponta a população de 600 indivíduos como a mais eficaz para esta implementação. Essa configuração alcançou o menor *fitness* médio (4,5) e demonstrou alta estabilidade, com o segundo menor desvio padrão (1,96). Populações maiores, como esta, mantêm a diversidade genética, o que reduz o risco de estagnação em ótimos locais e permite ao algoritmo encontrar soluções mais refinadas. Em contraste, populações menores tendem a convergir prematuramente para resultados inferiores. Assim, o tamanho de 600 indivíduos foi consolidado como o mais adequado, embora ainda não tivesse obtido a solução ótima.

4.5 Experimentos com diferentes métodos de mutação e recombinação

Após a definição de uma configuração de base sólida na etapa anterior, a pesquisa avançou para investigar se operadores genéticos alternativos poderiam elevar ainda mais o desempenho do algoritmo, uma vez que a solução ótima ainda não havia sido encontrada. O objetivo foi comparar essas novas funções contra os operadores padrão utilizados até então: a Mutação Uniforme e a Recombinação Uniforme. As funções escolhidas para esses experimentos foram:

- Mutação por troca: para cada indivíduo, caso a taxa de mutação seja atingida, altera-se o gene trocando os valores de duas posições aleatórias não fixas.
- Mutação por gene: para cada gene de cada indivíduo, verifica se ocorrerá mutação e, caso ocorra, substitui o valor da célula por outro sorteado aleatoriamente dentro do intervalo de 1 a 9.
- Recombinação de ponto único: gera um novo indivíduo herdando os genes do primeiro pai até um ponto de corte aleatório e os genes do segundo pai após esse ponto.

As taxas para os parâmetros definidos anteriormente (mutação = 1%, elitismo = 5%, torneio = 3 e população = 600) permaneceram inalterados. Os resultados foram consolidados na Tabela 7. Para melhor visualização da tabela, as palavras "Mutação" e "Recombinação" foram abreviadas para "M." e "R." respectivamente.

Tabela 7 – Resultados comparativos entre funções de mutação e recombinação.

Parâmetros	M. Uniforme e R. Uniforme	M. Uniforme e R. 1 Ponto	M. por Troca e R. Uniforme	M. por Gene e R. Uniforme
População	Melhor <i>fitness</i>	Melhor <i>fitness</i>	Melhor <i>fitness</i>	Melhor <i>fitness</i>
600	4	15	9	0
	7	9	4	2
	4	16	6	6
	8	19	4	6
	6	8	5	6
	4	17	2	5
	2	14	3	4
	2	14	5	8
	4	17	6	0
	4	7	2	5
Média	4,5	13,6	4,6	4,2
Desvio Padrão	1,96	4,17	2,12	2,70
Melhor	2	7	2	0
Pior	8	19	9	8

Fonte: A autora.

4.5.1 Análise dos resultados

A Tabela 7 foi estruturada para permitir uma análise, avaliando o impacto dos operadores de mutação e recombinação de forma isolada e comparativa. A análise, portanto, foi dividida em duas etapas.

- Análise do Método de Recombinação: para análise do método de recombinação, o método de mutação foi mantido constante (Uniforme). A Recombinação de 1 Ponto apresentou um desempenho significativamente inferior, com uma média de *fitness* muito alta (13,6). Em contraste, a **Recombinação Uniforme** se mostrou muito mais promissora (média de melhor *fitness* de 4,5), sendo assim consolidada como o método de recombinação padrão.
- Análise do Método de Mutação: o segundo passo foi comparar os métodos de mutação, agora com o método de recombinação constante (Uniforme). Nesta comparação, a **Mutação por Gene** não apenas alcançou a melhor média de *fitness* (4,2), como também, e mais importante, foi a única configuração a encontrar a solução ótima (*fitness* igual a 0), fazendo-o em duas das dez execuções. A Mutação por Troca, apesar de apresentar boa estabilidade (menor desvio padrão), não demonstrou o mesmo potencial para atingir a solução perfeita.

Após análise tanto dos métodos de recombinação quanto dos métodos de mutação, os operadores escolhidos foram Recombinação Uniforme e Mutação por Gene,

pelos seus desempenhos.

4.6 Experimentos comparando com a literatura

Para fins de comparação, foram feitos experimentos com e sem um método de pré-processamento, tomando como referência a literatura, especificamente o trabalho de [Neves \(2020\)](#), que avalia a eficiência de um AG também para resolução do Sudoku, comparando diferentes níveis de dificuldade.

4.6.1 Comparação dos AGs com pré-processamento

A metodologia para comparação do desempenho dos AGs com pré-processamento foi dividida em duas etapas:

- **Replicação da literatura:** Primeiramente, para estabelecer uma base de referência, realizou-se uma replicação fidedigna do algoritmo proposto por [Neves \(2020\)](#). Essa replicação incluiu sua função de pré-processamento lógico, que otimiza a população inicial analisando iterativamente as linhas, colunas e blocos de cada célula vazia. O processo preenche células que possuem um único candidato e se repete até que nenhuma alteração nova ocorra no tabuleiro.
- **Modelo proposto:** Em seguida, a abordagem proposta neste trabalho foi executada sob condições de teste análogas, para garantir a validade da análise comparativa.

Ambas as etapas são detalhadas nas subseções a seguir.

4.6.1.1 Replicação da literatura

A replicação do experimento de [Neves \(2020\)](#) seguiu as seguintes condições experimentais para os testes:

- Base de testes:
 - Foram utilizados 10 arquivos de entrada, distribuídos em 3 níveis de dificuldade (fácil, médio e difícil): 4 instâncias para o nível fácil (incluindo a matriz base utilizada na calibração do AG proposto) e 3 instâncias para cada um dos níveis médio e difícil.

- Com exceção da matriz usada na calibração, todos os arquivos foram obtidos do repositório público do autor⁹, e as matrizes correspondentes estão disponíveis no Anexo A.
- Parâmetros variáveis do AG:
 - Taxa de mutação: 4 níveis (30%, 50%, 70%, 90%).
 - Taxa de recombinação: 5 níveis (30%, 40%, 50%, 60%, 70%).
 - Tamanho da população: 25 níveis (de 4 a 100, com incremento de 4).
 - Repetições: cada combinação de parâmetros foi executada 4 vezes.
- Métodos do AG:
 - População inicial: gerada de forma aleatória sem repetição nas submatrizes.
 - Método de seleção: roleta.
 - Critério de parada: execução limitada a 100 gerações ou até a solução ótima (*fitness* = 0) ser encontrada.
 - Operador de mutação: mutação por troca nos blocos (submatrizes).
 - Operador de Recombinação: ponto único, nos blocos.

A combinação de todos os fatores resultou em um total de 20.000 execuções, conforme o cálculo: 10 arquivos x 4 taxas de mutação x 5 taxas de recombinação x 25 tamanhos de população x 4 repetições = 20.000.

4.6.1.2 Modelo proposto

Para uma análise comparativa justa, o modelo proposto foi avaliado sob condições experimentais análogas às da replicação, mantendo a mesma base de testes e os mesmos parâmetros variáveis.

As adaptações da taxa de recombinação e dos métodos do AG para este experimento foram:

- Taxa de recombinação: constante (100%).
- População inicial: gerada de forma puramente aleatória (sem a restrição de não repetição nos blocos).
- Método de seleção: roleta.
- Operador de mutação: por gene.

⁹<https://github.com/ifertz/SudokuSolver/tree/master>

- Operador de recombinação: uniforme.

O número total de execuções para o modelo proposto foi de 4.000, conforme o cálculo: 10 arquivos $\times$ 4 mutações $\times$ 1 recombinação $\times$ 25 populações $\times$ 4 repetições = 4.000.

As execuções descritas nas seções anteriores geraram dados de desempenho para ambas as abordagens. A Tabela 8 consolida esses resultados, comparando o modelo proposto com a abordagem replicada da literatura.

A análise é apresentada por matriz, destacando duas métricas principais: a quantidade de soluções ótimas (*fitness* igual a 0) encontradas e a taxa de sucesso para cada modelo.

Tabela 8 – Comparação de resultados por matriz com pré-processamento.

Modelo	Matriz	Ótimos encontrados	Porcentagem com relação ao total (%)
Replicação Neves (2020)	Matriz Base	2000	100
Modelo proposto		400	100
Replicação Neves (2020)	Matriz Fácil 1	2000	100
Modelo proposto		400	100
Replicação Neves (2020)	Matriz Fácil 2	2000	100
Modelo proposto		400	100
Replicação Neves (2020)	Matriz Fácil 3	2000	100
Modelo proposto		400	100
Replicação Neves (2020)	Matriz Médio 1	0	0
Modelo proposto		0	0
Replicação Neves (2020)	Matriz Médio 2	0	0
Modelo proposto		0	0
Replicação Neves (2020)	Matriz Médio 3	0	0
Modelo proposto		0	0
Replicação Neves (2020)	Matriz Difícil 1	0	0
Modelo proposto		0	0
Replicação Neves (2020)	Matriz Difícil 2	0	0
Modelo proposto		0	0
Replicação Neves (2020)	Matriz Difícil 3	0	0
Modelo proposto		0	0

4.6.1.3 Análise dos resultados

A análise detalhada dos resultados da Tabela 8 é apresentada a seguir, dividida por nível de dificuldade:

- Nível fácil: ambas as abordagens obtiveram 100%. A explicação reside na natureza dos dados de teste, pois os Sudokus de nível fácil são solucionáveis por

lógica determinística. Em cada etapa, sempre há pelo menos uma célula ou posição que pode ser preenchida sem ambiguidade (dentro de uma linha, coluna ou bloco). Como a função de pré-processamento implementa precisamente essa lógica, ela resolveu os tabuleiros de forma autônoma, tornando a atuação dos AGs desnecessária.

- **Nível médio:** o resultado foi de 0% para o modelo proposto e a replicação, não reproduzindo os 15,9% de êxito reportados por [Neves \(2020\)](#). Isso ocorre porque a lógica determinística do pré-processamento é insuficiente para solucionar quebra-cabeças de complexidade média, e os AGs também não foram capazes de encontrar a solução ótima posteriormente.
- **Nível difícil:** de forma análoga ao nível médio, o sucesso foi nulo (0%) para o modelo proposto e a replicação. Assim, o desempenho de 10,9% reportado por [Neves \(2020\)](#) também não foi obtido neste experimento. A justificativa é a mesma, porém mais acentuada: a eficácia do pré-processamento é ainda mais limitada pela maior complexidade dos tabuleiros, e os AGs tampouco demonstraram capacidade para resolver o problema de forma autônoma.

Em suma, neste experimento foram utilizadas as mesmas taxas de mutação (30%, 50%, 70% e 90%), os mesmos tamanhos de população (de 4 a 100, com incremento de 4), repetidos 4 vezes para cada configuração nos dois modelos. Os dois AGs obtiveram resultados similares (Tabela 8) mostrando que para os casos de nível fácil as soluções ótimas são obtidas integralmente pelo pré-processamento, e nenhuma das abordagens (proposta ou replicada) foi capaz de encontrar a solução ótima para os quebra-cabeças de complexidade média ou difícil com os parâmetros de [Neves \(2020\)](#).

Isto indica que o pré-processamento foi capaz de encontrar a solução ótima em níveis fáceis, justamente pela baixa complexidade das instâncias, mas quando se aumentava a dificuldade do problema, o AG não foi suficiente para resolver o problema utilizando estes parâmetros escolhidos. De fato, estas taxas de mutação estavam altas, e de acordo com a literatura, podiam trazer resultados insatisfatórios, bem como o tamanho de população, que chega, no máximo, a 100 indivíduos, diferente do que se concluiu empiricamente na Seção 4.4 que o tamanho adequado seria 600.

Para verificar se era o pré-processamento que resolvia o problema e avaliar a real contribuição dos AGs, o mesmo conjunto de testes foi repetido sem essa função auxiliar. O resultado foi nulo: nenhuma abordagem encontrou a solução, provando que o pré-processamento foi o único responsável pelo sucesso nos casos fáceis para estes experimentos. Além disto, a análise desses dados permitiu identificar a configuração

de maior potencial da replicação de [Neves \(2020\)](#) — aquela com os menores valores de *fitness*, que foi então selecionada para o teste comparativo a seguir.

4.6.2 Comparação dos AGs sem pré-processamento

A análise anterior (Seção 4.6.1.3) revelou que os dois AGs com pré-processamento resolveram o problema em 100% dos testes nos casos fáceis, enquanto nos cenários mais difíceis, mostraram-se ineficientes. Para preencher essa lacuna e medir a eficiência real de cada modelo, foi elaborado um experimento sem o auxílio do pré-processamento. Para este experimento também foram implementados: a **replicação da referência** e o **modelo proposto**. Este teste utilizou as melhores configurações de calibração na Matriz Base (nível fácil) e nas matrizes de todos os níveis de dificuldade (fácil, médio e difícil) de [Neves \(2020\)](#) para ambos os AGs.

4.6.2.1 Replicação da literatura

Com base nos experimentos feitos com a replicação do algoritmo de [Neves \(2020\)](#), a configuração que demonstrou maior potencial nos testes e foi selecionada para este comparativo foi a seguinte:

- Tamanho da população: 24 indivíduos.
- Seleção: roleta.
- Recombinação: ponto único, com taxa de 40%.
- Mutação: por troca, com taxa de 90%.
- Critério de parada: 100 gerações ou o encontro da solução ótima.
- População inicial: gerada de forma aleatória, sem repetição nas submatrizes.

4.6.2.2 Modelo proposto

A configuração que obteve melhor desempenho no modelo proposto, definida empiricamente nas seções anteriores, foi:

- Tamanho da população: 600 indivíduos.
- Seleção: torneio, com tamanho 3.
- Elitismo: taxa de 5%.

- Recombinação: uniforme, com taxa de 100%.
- Mutação: por gene, com taxa de 1%.
- Critério de parada: 200 gerações ou o encontro da solução ótima.
- População inicial: gerada de forma aleatória.

Cada configuração descrita foi executada 10 vezes para cada matriz de teste. Os resultados são apresentados na Tabela 9, na qual a “Matriz Base” refere-se à instância utilizada para a calibração do modelo proposto, e as demais correspondem às matrizes de nível fácil, médio e difícil do conjunto de testes de [Neves \(2020\)](#).

Tabela 9 – Comparação de resultados por matriz sem pré-processamento.

Modelo	Matriz	Ótimos encontrados	Porcentagem com relação ao total (%)
Replicação Neves (2020)	Matriz Base	0	0
Modelo proposto		2	20
Replicação Neves (2020)	Matriz Fácil 1	0	0
Modelo proposto		1	10
Replicação Neves (2020)	Matriz Fácil 2	0	0
Modelo proposto		0	0
Replicação Neves (2020)	Matriz Fácil 3	0	0
Modelo proposto		3	30
Replicação Neves (2020)	Matriz Médio 1	0	0
Modelo proposto		0	0
Replicação Neves (2020)	Matriz Médio 2	0	0
Modelo proposto		0	0
Replicação Neves (2020)	Matriz Médio 3	0	0
Modelo proposto		0	0
Replicação Neves (2020)	Matriz Difícil 1	0	0
Modelo proposto		0	0
Replicação Neves (2020)	Matriz Difícil 2	0	0
Modelo proposto		0	0
Replicação Neves (2020)	Matriz Difícil 3	0	0
Modelo proposto		0	0

4.6.2.3 Análise dos resultados

Este experimento foi realizado com as melhores configurações para cada modelo, sem pré-processamento, justamente para analisar a eficiência dos AGs sem nenhum método auxiliar.

Os resultados da Tabela 9 mostram a superioridade do modelo proposto, que foi o único capaz de encontrar soluções ótimas. O sucesso ocorreu nos quebra-cabeças

de menor complexidade, com o AG proposto atingindo 20% de êxito com a Matriz Base e obtendo sucesso em duas das três matrizes de nível fácil (Fácil 1 e Fácil 3, com 10% e 30% de êxito, respectivamente). O desempenho positivo do modelo proposto deve-se à sua fase de calibração dos parâmetros, confirmando os resultados obtidos nas Seções 4.1 a 4.5.

Em contrapartida, a réplica de [Neves \(2020\)](#) não convergiu em nenhuma das instâncias testadas sem pré-processamento, confirmando que os parâmetros escolhidos por ele não foram suficientes para o processo evolutivo de seu AG ocorrer do modo esperado.

Nos testes com as matrizes de nível médio e difícil, nenhuma das duas abordagens foi capaz de encontrar a solução ótima, mostrando uma necessidade de melhoria nas estratégias evolutivas de ambos os modelos para estes casos.

4.7 Considerações Finais

Este capítulo apresentou um processo metodológico detalhado para a calibragem dos parâmetros e dos operadores de um AG destinado à resolução do Sudoku. A partir de uma sequência de experimentos controlados, foi possível determinar empiricamente a configuração de maior eficácia. O processo iterativo de otimização culminou na seguinte parametrização final: população de 600 indivíduos, taxa de mutação de 1%, taxa de elitismo de 5% e seleção por torneio com tamanho 3. A combinação de Mutação por Gene e Recombinação Uniforme provou ser a abordagem mais eficaz, atingindo a solução ótima (*fitness* igual a 0) com maior regularidade e consolidando o resultado prático desta pesquisa.

A etapa de validação confrontou o modelo proposto com a réplica do algoritmo de [Neves \(2020\)](#). Um teste inicial em larga escala mostrou-se que a função de pré-processamento resolveu 100% dos casos fáceis, enquanto ambos os AGs falharam nos cenários de maior complexidade com o auxílio da heurística. Neste sentido, um segundo experimento foi realizado, comparando as melhores configurações dos dois AGs sem pré-processamento. Esta investigação demonstrou a superioridade do modelo desenvolvido neste trabalho em relação ao trabalho de referência, que atingiu taxas de sucesso de até 30% em instâncias fáceis enquanto a réplica de [Neves \(2020\)](#) não obteve nenhuma solução.

Em resumo, este trabalho definiu uma configuração adequada dos parâmetros do AG proposto para a resolução do problema do Sudoku no nível fácil, sem a utilização de nenhum método auxiliar, evidenciando a importância da etapa de ajuste desses parâmetros.

5 Conclusão

Este Trabalho de Conclusão de Curso teve como objetivo principal a aplicação de um AG para a resolução do Sudoku clássico, validando sua eficácia em instâncias nível fácil. A primeira contribuição fundamental deste estudo foi a determinação de uma configuração otimizada por meio de uma calibração metodológica rigorosa. Este processo demonstrou que o ajuste de parâmetros é o fator mais determinante para o desempenho e culminou em uma arquitetura robusta, com uma população de 600 indivíduos, taxa de mutação de 1%, taxa de elitismo de 5% e seleção por torneio com tamanho 3. A análise indicou que a escolha dos operadores genéticos desempenhou um papel fundamental, com a combinação da Recombinação Uniforme e da Mutação por Gene mostrando-se a abordagem mais eficaz para atingir a solução ótima com maior regularidade.

A segunda e principal contribuição deste estudo emergiu da análise comparativa, que avaliou a eficácia do AG em cenários com diferentes níveis de complexidade. Em testes com Sudokus de nível fácil, o estudo de [Neves \(2020\)](#) demonstrou que uma etapa de pré-processamento lógico foi suficiente para solucionar os problemas em 100% dos casos. A necessidade de um melhor ajuste dos parâmetros dos AGs foi evidenciada nos testes sem o método de pré-processamento, em que o AG desenvolvido neste trabalho conseguiu alcançar a solução ótima em até 30% dos casos fáceis, contrapondo-se ao AG de [Neves \(2020\)](#) que não obteve a solução ótima em nenhum dos testes sem pré-processamento. Isto indicou que as taxas escolhidas para seus operadores não foram adequadas para que o processo evolutivo ocorresse da melhor maneira. Estes experimentos confirmam a importância da etapa de calibração dos parâmetros do AG, evidenciando que ela é fundamental para a obtenção de soluções ótimas. Para as matrizes de nível médio e difícil, nenhum dos dois AGs foi capaz de encontrar a solução ótima, indicando que é necessário realizar melhorias para se obter um melhor resultado dos AGs nestes casos.

Para dar continuidade e superar as limitações identificadas, diversas frentes de pesquisa podem ser exploradas. Uma primeira melhoria seria remover a aleatoriedade da inicialização e da mutação, pois essa abordagem mostrou baixa eficiência, principalmente na versão totalmente aleatória aplicada. Em um nível mais avançado, sugere-se a evolução para um Algoritmo Memético, hibridizando o AG com uma técnica de busca local. Essa abordagem combinaria a capacidade de exploração global do AG com a otimização refinada da busca local, potencializando a velocidade de convergência e a capacidade de escapar de ótimos locais.

Referências

- BACK, T. **Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms**. New York: Oxford University Press, 1996. Citado na página 17.
- CORRÊA, R.; OBREGON, L. C.; MORAIS, A. P. de; MARIOTTO, L.; CARDOSO JR., G.; GUARDA, F. K. Coordenação de relés de sobrecorrente em sistemas de potência industriais: Uma abordagem considerando algoritmos genéticos. In: **9th IEEE/IAS International Conference on Industrial Applications, São Paulo**. [S.l.: s.n.], 2010. Citado na página 17.
- CUNHA, H. G. V. O. da. Algoritmo genético e algoritmo de vaga-lumes aplicados ao problema do caixeiro viajante. Universidade Federal de Uberlândia, 2019. Citado na página 18.
- DARWIN, C. **A origem das espécies**. Londres, Inglaterra, Reino Unido: John Murray, 1859. Citado 2 vezes nas páginas 13 e 16.
- DE JONG, K. A. **Evolutionary computation: a unified approach**. Cambridge, MA, EUA: MIT Press, 2006. Citado 2 vezes nas páginas 17 e 18.
- DELAHAYE, J.-P. The science behind sudoku. **Scientific American**, JSTOR, v. 294, n. 6, p. 80–87, 2006. Citado na página 19.
- DENG, X. Q.; LI, Y. D. A novel hybrid genetic algorithm for solving sudoku puzzles. **Optimization Letters**, Springer, v. 7, p. 241–257, 2013. Citado 3 vezes nas páginas 10, 13 e 26.
- DOMINGUES, C. V. B. **Aplicação de algoritmos genéticos em manobras espaciais**. Dissertação (Trabalho de Conclusão de Curso de Bacharelado) — Universidade Estadual Paulista (Unesp), São João da Boa Vista, SP, 2024. Citado na página 18.
- EIBEN, A. E.; SMITH, J. E. **Introduction to evolutionary computing**. 2. ed. Berlim e Heidelberg, Alemanha: Springer, 2015. Citado 2 vezes nas páginas 17 e 19.
- GABRIEL, P. H. R.; DELBEM, A. C. B. **Fundamentos de algoritmos evolutivos**. São Carlos: ICMC-USP, 2008. Citado na página 16.
- GAREY, M. R.; JOHNSON, D. S. **Computers and Intractability: A Guide to the Theory of NP-Completeness**. San Francisco: W. H. Freeman, 1979. Citado na página 13.
- GARNS, H. **Number Place**. 1979. Publicado na revista Dell Pencil Puzzles and Word Games. Considerado o precursor do Sudoku moderno. Citado na página 19.
- GOLDBERG, D. E. **Genetic algorithms in search, optimization, and machine learning**. Massachusetts, USA: Addison Wesley, 1989. Citado 2 vezes nas páginas 13 e 16.

GRANADOS, B. G.; M., C. G. Q.; NÚÑEZ, C. V. Improved genetic algorithm approach for coordinating decision-making in technological disaster management. **Neural Computing and Applications**, Springer, v. 36, n. 9, p. 4503–4521, 2024. Citado na página 18.

HOLLAND, J. H. **Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control and artificial intelligence**. Cambridge, MA, USA: MIT Press, 1975. Citado na página 16.

MANTERE, T.; KOLJONEN, J. Solving, rating and generating sudoku puzzles with ga. In: **2007 IEEE Congress on Evolutionary Computation**. Singapore: IEEE, 2007. p. 1382–1389. Citado 3 vezes nas páginas 10, 13 e 26.

MCCAFFREY, J. Execução de teste - resolução de sudoku usando a evolução combinatória. **MSDN Magazine**, Microsoft, v. 31, n. 11, p. N/A, nov 2016. Disponível em: <<https://learn.microsoft.com/pt-br/archive/msdn-magazine/2016/november/test-run-solving-sudoku-using-combinatorial-evolution>>. Citado na página 20.

NASCIMENTO, M. O.; BRASIL, C. R. S. Uma investigação sobre os operadores genéticos no algoritmo evolutivo aplicado ao problema do percurso do cavalo. **Revista Brasileira de Computação Aplicada**, v. 16, n. 3, p. 48–62, 2024. Citado na página 18.

NEVES, F. A. A. S. C. **Análise da eficiência de um algoritmo genético aplicado ao Sudoku**. Dissertação (Trabalho de Conclusão de Curso de Bacharelado) — Universidade Tecnológica Federal do Paraná, Cornélio Procopio, PR, 2020. Citado 14 vezes nas páginas 10, 13, 26, 27, 32, 33, 42, 44, 45, 46, 47, 48, 49 e 55.

OLIVEIRA, T. A. de. **Sudoku: Um estudo sobre algoritmos de geração, classificação e resolução para aplicação educacional**. Dissertação (Trabalho de Conclusão de Curso de Bacharelado) — Universidade Federal do Rio Grande do Sul, Porto Alegre, RS, 2024. Citado 6 vezes nas páginas 10, 13, 20, 21, 22 e 27.

OSTRAU, C.; KLARHORST, C.; THIES, M.; RÜCKERT, U. Benchmarking neuromorphic hardware and its energy expenditure. **Frontiers in Neuroscience**, Frontiers Media SA, v. 16, p. 873935, 2022. Citado na página 20.

SCHMITZ, G. C. Otimização de malha viária terrestre através de algoritmo genético. **Monografia (Bacharelado em Sistemas de Informação)—Sistemas de Informação, Universidade Federal de Santa Catarina, Florianópolis**, p. 5, 2022. Citado na página 18.

XU, L.; WANG, X.; WANG, Z.; CAO, G. Hybrid quantum genetic algorithm for structural damage identification. **Computer Methods in Applied Mechanics and Engineering**, Elsevier, v. 438, p. 117866, 2025. Citado na página 18.

YATO, T.; SETA, T. The complexity of the sudoku puzzle. **IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences**, E86-A, n. 5, p. 1052–1060, 2003. Citado na página 13.

Apêndices

APÊNDICE A – Matriz utilizada como base no desenvolvimento do AG

A seguir, a Figura 10 apresenta a matriz base utilizada no desenvolvimento do AG, correspondente a uma instância de nível fácil, em que as células com valor 0 representam as posições inicialmente vazias do Sudoku.

Figura 10 – Matriz base utilizada para testes.

0	8	4	0	7	2	1	0	5
2	0	7	8	3	0	9	0	0
6	0	0	5	0	9	0	0	8
0	6	0	9	2	8	4	0	0
0	7	0	0	0	0	0	6	9
0	2	0	0	0	0	0	8	1
0	3	2	0	5	0	6	9	4
7	0	0	0	0	0	0	0	2
1	0	0	2	0	4	0	0	7

Fonte: autora.

Anexos

ANEXO A – Matrizes utilizadas na comparação com a literatura (Seção 4.6)

A seguir, são apresentadas, nas Figuras 11 a 19, as nove matrizes de Sudoku de diferentes níveis, extraídas do repositório de Neves (2020)¹⁰, que serviram como base para os experimentos comparativos descritos na Seção 4.6, nas quais as células preenchidas com o número 0 representam os espaços inicialmente vazios.

Figuras 11, 12 e 13 - Matrizes de Sudoku nível fácil:

Figura 11 – Matriz Sudoku Fácil 1.

0	3	0	0	6	8	1	7	0
0	0	0	1	0	0	0	0	3
0	1	0	7	3	2	9	0	0
0	8	0	0	1	4	0	5	0
6	4	0	0	0	0	0	9	1
0	5	0	9	8	0	0	2	0
0	0	2	3	9	7	0	1	0
4	0	0	0	0	6	0	0	0
0	7	8	4	5	0	0	6	0

Fonte: autora.

Figura 12 – Matriz Sudoku Fácil 2.

0	0	3	0	2	0	6	0	0
9	0	0	3	0	5	0	0	1
0	0	1	8	0	6	4	0	0
0	0	8	1	0	2	9	0	0
7	0	0	0	0	0	0	0	8
0	0	6	7	0	8	2	0	0
0	0	2	6	0	9	5	0	0
8	0	0	2	0	3	0	0	9
0	0	5	0	1	0	3	0	0

Fonte: autora.

¹⁰ <<https://github.com/ifertz/SudokuSolver/tree/master>>

Figura 13 – Matriz Sudoku Fácil 3.

8	0	2	0	5	0	7	0	1
0	0	7	0	8	2	4	6	0
0	1	0	9	0	0	0	0	0
6	0	0	0	0	1	8	3	2
5	0	0	0	0	0	0	0	9
1	8	4	3	0	0	0	0	6
0	0	0	0	0	4	0	2	0
0	9	5	6	1	0	3	0	0
3	0	8	0	9	0	6	0	7

Fonte: autora.

Figuras 14, 15 e 16 - Matrizes de Sudoku nível médio:

Figura 14 – Matriz Sudoku Médio 1.

0	0	0	0	0	0	0	8	5
0	0	0	2	1	0	0	0	9
9	6	0	0	8	0	1	0	0
5	0	0	8	0	0	0	1	6
0	0	0	0	0	0	0	0	0
8	9	0	0	0	6	0	0	7
0	0	9	0	7	0	0	5	2
3	0	0	0	5	4	0	0	0
4	8	0	0	0	0	0	0	0

Fonte: autora.

Figura 15 – Matriz Sudoku Médio 2.

0	0	0	0	0	0	0	0	0
0	7	9	0	5	0	1	8	0
8	0	0	0	0	0	0	0	7
0	0	7	3	0	6	8	0	0
4	5	0	7	0	8	0	9	6
0	0	3	5	0	2	7	0	0
7	0	0	0	0	0	0	0	5
0	1	6	0	3	0	4	2	0
0	0	0	0	0	0	0	0	0

Fonte: autora.

Figura 16 – Matriz Sudoku Médio 3.

3	8	0	0	0	0	0	0	0
0	0	0	4	0	0	7	8	5
0	0	9	0	2	0	3	0	0
0	6	0	0	9	0	0	0	0
8	0	0	3	0	2	0	0	9
0	0	0	0	4	0	0	7	0
0	0	1	0	7	0	5	0	0
4	9	5	0	0	6	0	0	0
0	0	0	0	0	0	0	9	2

Fonte: autora.

Figuras 17, 18 e 19 - Matrizes de Sudoku nível difícil:

Figura 17 – Matriz Sudoku Difícil 1.

0	0	0	0	0	0	3	0	0
0	2	0	6	0	0	0	0	0
9	1	3	0	0	2	0	6	0
0	0	0	3	0	7	9	0	0
0	3	0	0	8	0	5	0	0
0	0	8	0	0	0	0	0	2
3	0	0	0	5	6	0	0	0
0	0	0	2	0	0	6	7	0
5	0	2	8	0	0	0	0	0

Fonte: autora.

Figura 18 – Matriz Sudoku Difícil 2.

0	0	0	0	0	3	0	1	7
0	1	5	0	0	9	0	0	8
0	6	0	0	0	0	0	0	0
1	0	0	0	0	7	0	0	0
0	0	9	0	0	0	2	0	0
0	0	0	5	0	0	0	0	4
0	0	0	0	0	0	0	2	0
5	0	0	6	0	0	3	4	0
3	4	0	2	0	0	0	0	0

Fonte: autora.

Figura 19 – Matriz Sudoku Difícil 3.

0	0	0	7	0	0	8	0	0
0	0	6	0	0	0	0	3	1
0	4	0	0	0	2	0	0	0
0	2	4	0	7	0	0	0	0
0	1	0	0	3	0	0	8	0
0	0	0	0	6	0	2	9	0
0	0	0	8	0	0	0	7	0
8	6	0	0	0	0	5	0	0
0	0	2	0	0	6	0	0	0

Fonte: autora.