Explorando operadores de um Algoritmo Genético aplicado ao jogo Sudoku

Christiane Regina Soares Brasil[1], **Andressa Oliveira Bernardes** [1]

[1]Faculdade de Computação – Universidade Federal de Uberlândia (UFU)
Uberlândia – MG – Brasil

christiane,andressa.bernardes@ufu.br

Abstract. *This work presents the development and evaluation of a Genetic Algorithm (GA) to solve classic Sudoku game, an NP combinatorial optimization problem, in order to improve the results obtained in the studies of [Bernardes 2025]. The GA was implemented in Python, with individuals represented by 9x9 matrices and a fitness function based on conflict minimization. The modeling of Sudoku as a permutation problem and the adopted methodology were decisive factors for the algorithm's success, surpassing the main reference (from 20% to 90% in the analyzed instance).*

Resumo. *Este trabalho apresenta o desenvolvimento e a avaliação de um Algoritmo Genético (AG) para resolver o jogo Sudoku clássico, um problema de otimização combinatória classificado como NP, a fim de melhorar os resultados obtidos nos estudos de [Bernardes 2025]. O AG foi implementado em Python, com indivíduos representados por matrizes 9x9 e com uma função de fitness baseada na minimização de conflitos. A modelagem do Sudoku como problema de permutação e a metodologia utilizada foram os fatores decisivos para o sucesso do algoritmo, superando a principal referência (de 20% para 90% na instância analisada).*

1. Introdução

Os Algoritmos Genéticos (AGs) [Holland 1975] são técnicas computacionais baseadas nos princípios da Teoria da Evolução [Darwin 1859], amplamente aplicadas em problemas complexos, em que abordagens determinísticas apresentam limitações para alcançar a solução ótima. Tem-se exemplos clássicos de otimização combinatória na literatura, tais como os problemas da Mochila [Bellman 1957, Dantzig 1957] e do Caixeiro Viajante [Flood 1956].

Neste contexto, o Sudoku é um quebra-cabeça lógico amplamente reconhecido como um problema combinatório de alta complexidade, apresentando diversas variações, como as versões: clássica, diagonal e *killer*, entre outras. Apesar de suas regras simples, o Sudoku é classificado como um problema NP (do inglês *Nondeterministic Polynomial time*)[1] [Yato and Seta 2003], o que implica uma elevada dificuldade computacional para encontrar soluções ótimas utilizando métodos de busca determinísticos. Essa característica torna necessária a aplicação de algoritmos capazes de explorar de forma mais

[1]Um 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 and Johnson 1979].

eficiente seu extenso espaço de busca, destacando os Algoritmos Evolutivos como uma alternativa promissora para sua resolução.

O Sudoku possui aplicações práticas em diferentes áreas, tais como escalonamento de tarefas e funcionários, planejamento de horários escolares, criptografia e verificação de integridade de dados, tornando-o um problema ainda atual e relevante a ser estudado.

O objetivo deste trabalho foi desenvolver e otimizar um Algoritmo Genético para a resolução do Sudoku clássico, a fim de melhorar os resultados da pesquisa de [Bernardes 2025] por meio de operadores específicos para representações permutacionais [Eiben and Smith 2015], como a mutação por troca e a recombinação OX. Para tal, foram analisados diferentes configurações de parâmetros e operadores genéticos que podem influenciar o desempenho do algoritmo, focando na taxa de sucesso em obter a solução ótima.

Na próxima seção, o jogo Sudoku clássico será descrito, com suas respectivas regras.

2. O jogo de tabuleiro: Sudoku

O Sudoku clássico teve origem no jogo *Number Place* [Garns 1979] e consiste em uma grade 9x9 dividida em nove subgrades 3x3, em que o desafio é preencher os espaços vazios com números, seguindo as seguintes regras: não pode haver repetições nas linhas; não pode haver repetições nas colunas; e não pode haver repetições nas submatrizes. O jogo é vencido quando todos os números são encaixados em suas respectivas células. A quantidade de valores já fornecidos no início varia conforme o nível de dificuldade, ou seja, quanto mais difícil, menos números são dados na configuração inicial [Bernardes 2025].

A utilização do Sudoku como tema de estudos na computação vai além dos algoritmos desenvolvidos para resolução do problema em questão, sendo amplamente empregado como *benchmark* em pesquisas de alta complexidade combinatorial. O Sudoku possui aplicações reais em áreas como otimização, inteligência artificial, teoria dos grafos e satisfação de restrições, sendo importante como modelo para o desenvolvimento e avaliação de algoritmos capazes de resolver problemas combinatórios complexos.

Na próxima seção, será apresentada a metodologia para a elaboração deste AG aplicado ao problema do Sudoku clássico.

3. Metodologia

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 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/artigo

A seguir, será apresentada a descrição do Algoritmo Genético desenvolvido para resolver o Sudoku clássico, incluindo: a representação do indivíduo, a geração da população inicial, a função de *fitness* e os operadores genéticos que atuam neste AG (seleção, recombinação, mutação e elitismo).

3.1. Representação do Indivíduo

Cada indivíduo representou uma solução candidata para o Sudoku e foi implementado como uma classe com dois atributos: uma matriz 9x9 e um valor de *fitness* que avaliou a qualidade da solução. As células com números fixos (predispostos) do tabuleiro permaneceram inalteradas durante toda a evolução, enquanto as células vazias, representadas por zeros, foram preenchidas e modificadas pelos operadores genéticos a cada geração [Bernardes 2025].

3.2. Geração da População Inicial

A população inicial corresponde ao primeiro conjunto de indivíduos do Algoritmo Genético. Neste trabalho, cada indivíduo foi gerado aleatoriamente, garantindo que não houvesse repetição de números nas linhas. Durante o preenchimento, os números já utilizados foram armazenados para evitar duplicações, garantindo desde o início o cumprimento de uma das principais restrições do Sudoku.

3.3. Função de *Fitness*

A função de *fitness* avaliou a qualidade de cada indivíduo em relação ao objetivo do problema, medindo o quão próximo ele está de ser uma solução válida para o Sudoku [Bernardes 2025]. Esse valor foi calculado com base na soma total das violações das regras do jogo, funcionando da seguinte maneira: uma variável de contagem de conflitos foi inicializada com zero; em seguida, o algoritmo percorreu as 9 colunas da matriz, contando os números repetidos e somando as repetições ao contador; por fim, o mesmo procedimento foi aplicado às 9 submatrizes 3x3.

Nesta implementação, não houve verificação nas linhas, uma vez que a geração inicial garantiu a não repetição de números nelas. O valor retornado representou o total de conflitos. Deste modo, um *fitness* igual a 0 indica uma solução perfeita, sem violações das regras do Sudoku. Portanto, o objetivo deste AG foi minimizar esse valor até alcançar zero.

3.4. Operadores Genéticos

Neste trabalho, foi utilizado o método de seleção por torneio, no qual grupos de indivíduos competem e os mais aptos são escolhidos para reprodução [Eiben and Smith 2015, Bernardes 2025]. Para a recombinação, adotou-se o operador OX (*Order Crossover*) [Eiben and Smith 2015], aplicado linha a linha, preservando a não repetição de números. A mutação utilizada foi a mutação por troca [Eiben and Smith 2015, Neves 2020], que consiste em trocar valores entre posições não fixas de uma linha, contribuindo para a variabilidade genética e evitando ótimos locais. Além disso, foi implementado o elitismo [Bäck et al. 2000, Goldberg 1989], em que os indivíduos com melhor *fitness* são preservados diretamente para a próxima geração, sem sofrer alterações genéticas.

Deste modo, os métodos deste AG específico para o problema do Sudoku foram elaborados como descritos nesta seção e experimentados, de acordo com a descrição apresentada na próxima seção.

4. Experimentos e resultados

Esta seção apresenta a análise dos resultados obtidos com a aplicação do Algoritmo Genético ao problema do Sudoku clássico 9x9. Para avaliar a influência dos parâmetros

mutação, torneio e elitismo, todos os demais parâmetros foram mantidos fixos e definidos empiricamente: 10 execuções para cada configuração; critério de parada de 200 gerações ou obtenção da solução ótima (*fitness* igual a 0); população de 600 indivíduos; e utilização de uma instância de nível fácil [Bernardes 2025].

Deste modo, foram realizados diversos experimentos para definir as taxas de mutação, elitismo e o tamanho do torneio. Para isso, foram avaliadas taxas de mutação de 0,25%, 0,5%, 2% e 10%, taxas de elitismo de 5%, 10%, 15% e 20%, e tamanhos de torneio iguais a 3, 4 e 5. Com base na taxa de sucesso na obtenção da solução ótima, a configuração definida como padrão para este AG foi: taxa de mutação de 0,25%, taxa de elitismo de 5% e torneio com tamanho 4.

5. Comparação com a literatura

Nesta pesquisa, foi proposta uma extensão de [Bernardes 2025], incorporando operadores específicos para representações permutacionais [Eiben and Smith 2015], como mutação por troca, recombinação OX e geração inicial sem repetição nas linhas. Diferentemente de [Neves 2020] e [Bernardes 2025], que utilizaram outros operadores genéticos, a abordagem proposta elevou a taxa de sucesso de 20% para 90% e superou os resultados de [Neves 2020], que não obteve sucesso na resolução do Sudoku. Os resultados evidenciam a importância da modelagem baseada em permutação e da escolha adequada dos operadores genéticos para o desempenho do algoritmo.

6. Conclusão

Este trabalho desenvolveu e otimizou um Algoritmo Evolutivo para resolução do Sudoku clássico 9×9, dando continuidade ao estudo de [Bernardes 2025]. A principal contribuição em relação à referência principal foi a adoção de uma modelagem baseada em permutação, garantindo desde a inicialização a não repetição de números nas linhas, além da utilização da mutação por troca e da recombinação OX, operadores adequados a esse tipo de representação [Eiben and Smith 2015]. Essas escolhas reduziram a geração de indivíduos inválidos e direcionaram a busca para as restrições de colunas e grades. Os resultados demonstraram aumento da taxa de sucesso de 20% para 90% na instância fácil analisada, utilizando população de 600 indivíduos, torneio de tamanho 4, 200 gerações, 0,25% de mutação e 5% de elitismo, evidenciando a eficácia da modelagem e das adaptações propostas. Como trabalhos futuros, sugere-se avaliar o método em Sudokus mais complexos e incorporar técnicas de busca local inspiradas em [Wang et al. 2023] para aprimorar a convergência do algoritmo e possibilitar a aplicação do mesmo em instâncias maiores e mais complexas.

7. Agradecimentos

As autoras agradecem à Faculdade Computação da Universidade Federal de Uberlândia (Campus Santa Mônica) pelo apoio institucional fornecido para o desenvolvimento desta pesquisa.

Referências

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press. `https://dl.acm.org/doi/book/10.5555/1893145`.

Bernardes, A. O. (2025). Algoritmo genético aplicado ao jogo de tabuleiro sudoku. Trabalho de conclusão de curso de bacharelado, Universidade Federal de Uberlândia, Uberlândia, MG. https://repositorio.ufu.br/handle/123456789/47075.

Bäck, T., Fogel, D. B., and Michalewicz, Z. (2000). *Evolutionary Computation 1: Basic Algorithms and Operators*. Institute of Physics Publishing, Bristol, UK. https://dl.acm.org/doi/10.5555/553038.

Dantzig, G. B. (1957). Discrete-variable extremum problems. *Operations Research*, 5(2):266–288. https://doi.org/10.1287/opre.5.2.266.

Darwin, C. (1859). *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. John Murray, London, England, United Kingdom. https://psycnet.apa.org/doi/10.1037/14088-000.

Eiben, A. E. and Smith, J. E. (2015). *Introduction to evolutionary computing*. Springer, Berlim e Heidelberg, Alemanha, 2 edition. https://dl.acm.org/doi/10.5555/1965441.

Flood, M. M. (1956). The traveling-salesman problem. *Operations Research*, 4(1):61–75. https://doi.org/10.1287/opre.4.1.61.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, San Francisco. https://dl.acm.org/doi/book/10.5555/574848.

Garns, H. (1979). Number place. Publicado na revista Dell Pencil Puzzles and Word Games. Considerado o precursor do Sudoku moderno.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Addison Wesley, Massachusetts, USA. https://dl.acm.org/doi/10.5555/534133.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control and artificial intelligence*. MIT Press, Cambridge, MA, USA. https://doi.org/10.7551/mitpress/1090.001.0001.

Neves, F. A. A. S. C. (2020). Análise da eficiência de um algoritmo genético aplicado ao sudoku. Trabalho de conclusão de curso de bacharelado, Universidade Tecnológica Federal do Paraná, Cornélio Procópio, PR. http://repositorio.utfpr.edu.br/jspui/handle/1/28584.

Wang, C., Sun, B., Du, K.-J., Li, J.-Y., Zhan, Z.-H., Jeon, S.-W., Wang, H., and Zhang, J. (2023). A novel evolutionary algorithm with column and sub-block local search for sudoku puzzles. *IEEE Transactions on Games*, PP:1–11. https://doi.org/10.1109/TG.2023.3236490.

Yato, T. and Seta, T. (2003). Complexity and completeness of finding another solution and its application to puzzles. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E86-A(5):1052–1060.