Criptografia e Heurística: uma Implementação de Algoritmo Genético na Decriptação de Textos

Everson Bacelli

Resumo – Este trabalho teve como objetivo apresentar um algoritmo voltado à decifração de mensagens encriptadas por meio da técnica de substituição monoalfabética. A implementação foi desenvolvida em linguagem Python, utilizando um Algoritmo Genético (AG) como base heurística. Os resultados obtidos demonstram que o AG foi eficiente na decriptação dos textos-alvo na maioria das tentativas, além de permitir reflexões relevantes sobre os aspectos internos do método, como o comportamento evolutivo, a calibragem de parâmetros e o consumo de recursos computacionais. O estudo evidencia tanto os benefícios quanto as limitações da aplicação do AG nesse contexto.

Palavras-chave: Criptografia, Decriptação, Algoritmo Genético, Python.

Abstract – This work aimed to present an algorithm for decrypting messages encrypted using a monoalphabetic substitution cipher. The implementation was developed in Python based on a Genetic Algorithm (GA). The results indicate that the GA was effective in decrypting the target texts in most cases and provided valuable insights into its internal mechanisms, such as parameter tuning, evolutionary behavior, and computational resource consumption. The study highlights both the advantages and the limitations of using GAs in this context.

Keywords: Cryptography, Decryption, Genetic Algorithm, Python.

I. Introdução

Este trabalho foi desenvolvido no contexto do curso de Mestrado em Ciência da Computação da Universidade Federal do ABC (UFABC), na disciplina de Arquitetura de Computadores. Seu principal objetivo foi propor e implementar um algoritmo computacional capaz de decriptar mensagens encriptadas por meio da Cifra de Substituição Monoalfabética.

A criptografia é um campo essencial na atualidade, considerando a crescente dependência de sistemas computacionais que manipulam grandes volumes de informações. Sem o uso de técnicas criptográficas, muitos desses sistemas se tornariam inviáveis, pois a exposição de dados sensíveis pode comprometer a privacidade, a imagem e até mesmo a segurança de indivíduos e instituições.

O estudo da criptografia também oferece a oportunidade de compreender estratégias de resolução de problemas complexos, caracterizados por vastos espaços de busca e múltiplas combinações possíveis — cenário típico dos processos de criptoanálise. Essa investigação impõe desafios técnicos, como o uso de estruturas de dados e algoritmos eficientes, e analíticos, relacionados à avaliação das características de cada técnica de encriptação.

A implementação prática deste projeto foi particularmente relevante por permitir uma exploração aprofundada da estratégia escolhida — o Algoritmo Genético. Essa técnica possibilitou a análise de suas principais operações (seleção, cruzamento e mutação) e de seu impacto sobre o desempenho computacional. Para fins de organização, este relatório está estruturado da seguinte forma, além desta introdução:

- Seção II – Referencial Teórico: apresenta os conceitos fundamentais relacionados à criptografia e aos algoritmos genéticos;
- Seção III – Metodologia: descreve o método desenvolvido para a quebra da cifra, com ênfase na implementação do AG;
- Seção IV – Resultados: analisa os dados coletados e a eficácia da abordagem;
- Seção V – Conclusões: discute as considerações finais e limitações da solução proposta.

II. Referencial Teórico

Este capítulo tem o objetivo apresentar os conceitos essenciais, expressões e técnicas presentes no trabalho, especialmente sobre Criptografia e Algoritmo Genético (AG).

2.1 Criptografia

A Criptologia é a ciência dedicada tanto à ocultação de informações quanto à sua revelação. De modo geral, pode ser definida como a arte de cifrar, codificar ou esconder dados, tornando-os incompreensíveis a leitores não

autorizados. Apenas receptores devidamente autenticados podem acessar o conteúdo original da mensagem [1].

Historicamente, o uso da criptografia surgiu da necessidade de proteger informações diplomáticas e militares. Com o avanço da tecnologia e o aumento do valor estratégico das informações, a criptografia evoluiu de uma arte empírica para uma ciência formal, tornando-se fundamental para a segurança da comunicação digital [2]. Hoje, seu papel é indispensável em diversas áreas, como comércio eletrônico, operações bancárias, troca de documentos eletrônicos, sigilo hospitalar e segurança militar [3].

2.1.1 Técnicas de Encriptação e Criptoanálise

As técnicas de encriptação transformam o texto claro (legível) em texto cifrado (ininteligível), baseando-se em dois princípios: substituição, que troca elementos, e transposição, que rearranja sua posição. Algoritmos modernos combinam ambos os métodos, sendo classificados como simétricos (mesma chave para cifrar e decifrar) ou assimétricos (par de chaves pública e privada) [4].

A quebra de criptografia, conhecida como criptoanálise, busca deduzir o texto claro ou a chave secreta. Os ataques podem ser de dois tipos principais: analíticos, que exploram características estatísticas da linguagem, e força bruta, que testa todas as chaves possíveis. No entanto, chaves de grande tamanho (como 128 bits) tornam o ataque por força bruta computacionalmente inviável [4].

2.1.2 Cifra de Substituição Monoalfabética

A cifra de substituição monoalfabética substitui cada letra do texto original por outra letra, número ou símbolo. Embora simples, essa técnica pode ser quebrada por meio de análise de frequência, uma vez que mantém padrões estatísticos da língua [1].
Ainda assim, o número de combinações possíveis (26!) a torna resistente a ataques de força bruta [4].

2.2 Algoritmos Genéticos

Os Algoritmos Genéticos (AGs) são técnicas de otimização e busca inspiradas nos princípios da evolução natural, conforme propostos por Charles Darwin [5]. Eles pertencem ao campo da Computação Evolutiva, sendo classificados como meta-heurísticas adaptativas, capazes de encontrar soluções aproximadas para problemas complexos.

Formalmente propostos por John Holland (1975) em *Adaptation in Natural and Artificial Systems*, os AGs simulam a evolução de uma população de soluções por meio da seleção, cruzamento (crossover) e mutação [7].

2.2.1 Funcionamento

Cada solução candidata (indivíduo) é representada por um cromossomo, que codifica parâmetros ou combinações possíveis. A população inicial é gerada aleatoriamente e avaliada por uma função de aptidão (fitness), que mede a qualidade de cada indivíduo.
Os mais aptos são selecionados para gerar novos indivíduos, combinando suas informações genéticas (crossover) e sofrendo pequenas alterações (mutação) para manter a diversidade. O processo evolui iterativamente até atingir um critério de parada, como o número máximo de gerações.

2.2.2 Vantagens e Limitações

Os AGs oferecem diversas vantagens: simplicidade de formulação, capacidade de explorar grandes espaços de busca e potencial para evitar mínimos locais [6]. Contudo, sua eficácia depende fortemente da calibragem dos parâmetros — como taxas de mutação e cruzamento — e de estratégias de elitismo e diversidade [5].
Uma parametrização inadequada pode levar à convergência prematura ou ao aumento do custo computacional. Além disso, como meta-heurística, o AG não garante a obtenção da solução ótima global [9].

III. Metodologia

Esta seção descreve a estrutura da solução desenvolvida em linguagem Python, com ênfase no processo de decriptação heurística baseado em um Algoritmo Genético (AG). O processo geral é representado em um diagrama BPMN 2.0, apresentado no Apêndice 1, que ilustra os principais estágios da execução.

O primeiro estágio teve como objetivo obter os binários dos textos encriptados. Para isso, utilizou-se o método `open()` do Python, que permitiu a leitura do conteúdo em formato binário e seu armazenamento em um vetor. Posteriormente, os binários foram convertidos em caracteres com base na tabela ASCII, ignorando espaços, pontuação e caracteres especiais. O resultado foi organizado em um vetor de palavras encriptadas, seguindo a lógica da Cifra de Substituição Monoalfabética.

A seguir, foi criada a primeira geração de soluções. Esse processo envolveu a identificação de todas as letras únicas presentes no texto, facilitando a visualização dos caracteres a serem substituídos e daqueles a serem ignorados. Para isso, foi implementado um modelo denominado Alternativa, no qual cada instância dessa classe continha uma combinação de chaves e valores

representando pares de substituição. Além disso, cada instância armazenava um valor de scoreFitness, indicador responsável por quantificar o grau de proximidade da solução em relação ao texto original.

Figura 1 - Estrutura da Chave - Cromossomo do modelo

[(‘A’, ‘x’), (‘B’, ‘p’), (‘C’, ‘x’), (‘D’, ‘m’) …]

Fonte: Elaborado pelo autor

Essa estrutura evidencia a semelhança conceitual entre o modelo proposto e o funcionamento biológico dos **cromossomos**, reforçando a aderência da implementação aos princípios fundamentais dos AGs.

Figura 2 - Estrutura de um Cromossomo com às Chaves e Valores

Fonte: Desenvolvido pelo Gemini com base em Prompt requisitado pelo autor.

A primeira geração foi composta por 50 instâncias da classe Alternativa, submetidas a até 5.000 iterações evolutivas. Em cada ciclo, 40 combinações entre as duas melhores alternativas eram selecionadas de forma aleatória para realizar o cruzamento (crossover) de seus genes. Posteriormente, os filhos eram submetidos a uma taxa de mutação de 0,4, promovendo diversidade genética na população.

Ao final de cada geração, as oito melhores alternativas eram preservadas (elitismo) e propagavam seu material genético para a geração seguinte, juntamente com algumas alternativas de menor *scoreFitness*, com o objetivo de manter a variabilidade populacional.

3.2.1 - Detalhes do Elementos Essenciais do AG presentes na aplicação:

A - Parâmetros Globais do Modelo

Os valores dos parâmetros utilizados foram definidos de forma heurística, com base em experimentação e análise dos resultados observados. A Tabela 1 apresenta os principais valores adotados.

Tabela 1 - Macro Parâmetros do Modelo

N° de Gerações	10.000
Tamanho da Geração Inicial	200
Taxa de mutação	0.4
Número de Mutações	40
Número de Sobreviventes	15

Fonte: Elaborado pelo autor

3.2 - ScoreFitness

A função *scoreFitness* foi desenvolvida para quantificar o desempenho de cada alternativa e identificar aquelas mais próximas da solução correta [9]. Ela combina dois componentes principais:

1. Log-Likelihood Score: avalia a presença e frequência de bi e trigramas (pares e trincas de letras) em comparação a listas estatísticas da língua inglesa, penalizando ocorrências inexistentes com um valor fixo (-10).

2. Frequência de Palavras Corretas (fpc): mede o número de palavras válidas no texto decifrado com base em um dicionário de termos da língua inglesa.

A pontuação final é obtida por:

$$ll\_score\_total=(0{,}25\times score\_bi)+(0{,}75\times score\_tri)$$

$$scoreFitness=ll\_score\_total+(fpc\times 2000)$$

Na prática, quanto mais palavras corretas e padrões linguísticos válidos o texto apresentar, **maior será o scoreFitness**, indicando uma solução mais próxima da chave correta.

3.3 - Seleção dos Melhores (Elitismo)

A seleção foi baseada no método de torneio, com tamanho de amostra igual a 3. Em cada torneio, três indivíduos eram escolhidos aleatoriamente, e o de melhor *scoreFitness* era selecionado para reprodução. Essa estratégia de elitismo garante que os indivíduos mais aptos transmitam seus genes à próxima geração, acelerando a convergência do algoritmo. [9].

3.4 - Estratégias de Mutação

A mutação desempenhou um papel essencial na manutenção da diversidade genética e na prevenção da convergência prematura. Duas estratégias principais foram utilizadas:

- Cruzamento Genético (Crossover): troca parcial dos genes entre dois indivíduos, com ponto e comprimento de corte definidos aleatoriamente.

- Mutação Direta: substituição aleatória de pares de chave-valor, evitando repetições e incentivando novas combinações [6,9].

3.5 - Estratégias de Controle de Convergência Prematura:

Em algumas execuções, o algoritmo apresentava convergência prematura, ou seja, atingia uma solução de alto score que se mantinha inalterada ao longo das

gerações. Para mitigar esse problema, foram adotadas as seguintes estratégias:

- Inclusão aleatória de soluções antigas em novas gerações, especialmente quando o *scoreFitness* permanecia estável por mais de cinco iterações;
- Inserção intencional de alternativas com baixo score, promovendo diversidade;
- Exclusão temporária da melhor solução quando seu score se repetia por cinco gerações consecutivas, forçando o sistema a explorar novas regiões do espaço de busca.

IV. Resultados

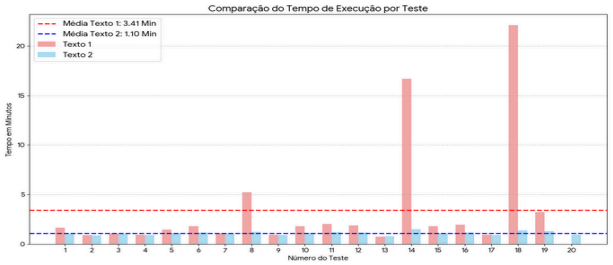
Os testes realizados compreenderam 20 execuções do algoritmo, cada uma configurada com os mesmos parâmetros evolutivos e variação apenas na semente aleatória. O objetivo foi avaliar a eficiência e a estabilidade do Algoritmo Genético (AG) em termos de tempo de execução e número de ciclos até a convergência.

Em 37 das 40 cerca de 95% das execuções o objetivo foi alcançado integralmente, resultando em uma decriptação completa do texto cifrado. Em 3 execuções, um pouco menos de 10%, o AG não convergiu para a solução ideal dentro do limite de iterações.

Também é possível visualizar testes que apresentaram dados bastante discrepantes tanto em relação ao tempo quanto ao número de iterações (ciclos), embora sejam a minoria, os gráficos 1 e 2 é possível visualizar isso. Desconsiderando tais casos, o desempenho médio do Texto 1 foi de 1.167 ciclos, contra 82 ciclos no Texto 2, indicando eficiência crescente com o aumento do texto.

O tempo médio de execução dos testes foi de aproximadamente 10,5 minutos, com variação entre 1,43 minutos e 44,74 minutos. O desvio padrão de 11,8 minutos indica uma dispersão moderada, associada principalmente à aleatoriedade da inicialização e à variabilidade no processo de mutação.

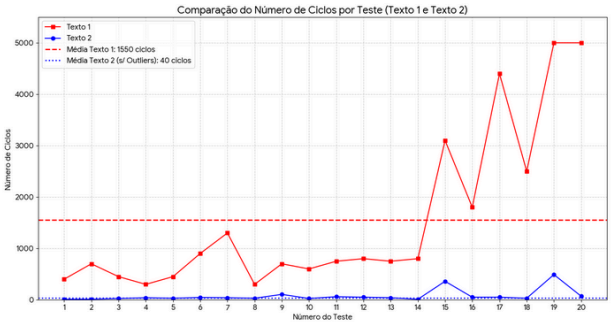
Gráfico 1 - Resultado dos testes aplicados aos dois textos em relação ao tempo de execução



Fonte: Desenvolvido pelo Autor

Verifica-se que a maior parte das execuções concentrou-se entre 3 e 8 minutos, enquanto os valores acima de 30 minutos correspondem a casos isolados de convergência tardia. Assim, de modo geral, o desempenho temporal do AG foi satisfatório, com predominância de execuções rápidas e comportamento estável, conforme pode ser observado no gráfico 1.

Gráfico 2 - Resultado dos testes aplicados aos dois textos em relação ao número de ciclos



Fonte: Desenvolvido pelo Autor

Em relação ao número de ciclos até a convergência apresentou comportamento semelhante ao tempo total, reforçando a correlação entre ambos. O valor mínimo observado foi de 15 ciclos, e o máximo, 4.470 ciclos, com uma média de 1.243 ciclos e desvio padrão de 1.186.

Portanto, Observa-se que o Texto 2, apesar de ser substancialmente maior, apresentou menor tempo médio e número de ciclos até a convergência. Essa característica demonstra que o AG se beneficia de um corpus mais rico, uma vez que a função de aptidão (scoreFitness) dispõe de mais informações linguísticas para diferenciar soluções promissoras. Em contrapartida, o Texto 1, com apenas 79 palavras, mostrou maior variabilidade entre execuções. O pequeno número de padrões linguísticos (bigrama e trigramas) reduz a precisão estatística, levando o AG a explorar mais gerações antes de atingir a chave correta.

V. Conclusões

Este trabalho teve como propósito desenvolver e analisar uma solução baseada em Algoritmos Genéticos para a decriptação de mensagens cifradas por substituição monoalfabética.

A proposta foi implementada com sucesso em Python, apresentando resultados satisfatórios quanto à taxa de acerto e eficiência.

O modelo Alternativa, concebido para representar os cromossomos, mostrou-se adequado para o processo evolutivo. A seleção por torneio, a mutação e o elitismo contribuíram para a robustez da busca, permitindo que o algoritmo atingisse resultados consistentes em diferentes execuções.

A taxa de sucesso de aproximadamente 95%, aliada ao baixo tempo de execução, demonstra o potencial da abordagem. Entretanto, algumas limitações devem ser destacadas:

- O experimento foi restrito a apenas dois textos; ampliar o conjunto de testes pode validar a generalização do modelo.
- A calibração dos parâmetros foi feita empiricamente; estudos mais rigorosos poderiam otimizar a taxa de mutação, número de gerações e tamanho da população.
- O uso de dicionários mais extensos melhora a acurácia, mas aumenta o tempo de processamento, configurando um trade-off relevante entre desempenho e precisão.

Por fim, o trabalho cumpriu seus objetivos, demonstrando a aplicabilidade dos Algoritmos Genéticos na resolução de problemas de criptoanálise. Além de contribuir para o entendimento da relação entre heurísticas e criptografia, abre espaço para pesquisas futuras sobre otimização adaptativa e técnicas híbridas envolvendo inteligência artificial e segurança da informação.

VI. Referências

[1] MENEZES NETO, José Laudelino de. Primeiros passos em criptografia. João Pessoa: Editora UFPB, 2021.

[2] SILVA, Willian Wallace de Matteus. A evolução da criptografia e suas técnicas ao longo da história. 2019. 27 p. Monografia (Bacharelado em Sistemas de Informação) - Instituto Federal Goiano, Campus Ceres, Ceres, 2019.

[3] MAZIERO, Carlos Alberto. *Sistemas Operacionais: Conceitos e Mecanismos*. Curitiba: DINF – Universidade Federal do Paraná, 2019. ISBN 978-85-7335-340-2. Disponível em: https://wiki.inf.ufpr.br/maziero/lib/exe/fetch.php?media=socm%3Asocm-00.pdf.

[4] STALLINGS, William. Criptografia e segurança de redes: princípios e práticas. Tradução Daniel Vieira; revisão técnica Paulo Sérgio Licciardi Messeder Barreto, Rafael Misoczki. 6. ed. São Paulo: Pearson Education do Brasil, 2015.

[5] PINTO, Anderson Rogério Faria; MARTARELLI, Nadia J.; NAGANO, Marcelo Seido. Algoritmo Genético: Principais Gaps, Trade-offs e Perspectivas para Futuras Pesquisas. Trends Comput. Appl. Math., v. 23, n. 3, 2022. DOI: 10.5540/tcam.2022.023.03.00413.

[6] IKEDA, Patrícia Akemi. Introdução aos Algoritmos Genéticos. São Paulo: Instituto de Matemática e Estatística – Universidade de São Paulo, nov. 2009. Disponível em: https://www.ime.usp.br/~gold/cursos/2009/mac5758/PatriciaGenetico.pdf.

[7] GONÇALVES, André R. Algoritmos Genéticos. [S.l.: s.n.], 2008. Disponível em: https://andreric.github.io/files/pdfs/geneticos.pdf

[8] TSURUTA, Jaime Hidehiko; NARCISO, Marcelo Gonçalves. Um estudo sobre algoritmos genéticos. Campinas, SP: Embrapa Informática Agropecuária, 2000. ISSN 1518-3602.

[9] KATO, Rodrigo; PAIVA, Vinícius; IZIDORO, Sandro. Algoritmos Genéticos. BIOINFO – Revista Brasileira de Bioinformática, Edição #01, jul. 2021. DOI: 10.51780/978-6-599-275326-13.

Apêndice 1 - Fluxograma do Algoritmo