COSC 3P71 ASSN2

Carson Cormier
COSC 3P71: Artificial Intelligence
Brock University
Thorold, Canada
ww23iu@brocku.ca — 7843469

***Abstract*—This report outlines the use of a Genetic Algorithm (GA) to break the Vigenere cipher. The GA uses a character-based chromosome, tournament selection, uniform crossover, and one-point crossover to conduct these tests. The experiments tested different crossover and mutation rates, analyzing how the Vigenere cipher reacts. The performance of the tests was measured by examining the average best fitness and average population fitness across multiple runs. Results showed that uniform crossover with a 1% mutation rate produced the best solutions, giving an average best fitness of 0.0859. The choice of crossover operator was analyzed to have a significant impact on the algorithm's performance.**

I. Introduction

This document reports on the implementation of a Genetic Algorithm (GA) to decrypt text encrypted with the Vigenere cipher. This Cryptanalysis is an important field for testing and improving encryption security in the real world. This project demonstrates how the GA attempts to discover the cipher key, showing how evolutionary algorithms can solve complex search problems unrecognizable by humans. This report will introduce the concepts, define the problem, and explain the importance of the solution.

II. Background

A Genetic Algorithm is a search heuristic inspired by natural selection found in biology. It essentially tries different solutions, keeps the better ones, and mixes them to create even better results. The basic procedure of the GA goes as follows:
- Read the problem data.
- Set the GA parameters (Such as crossover rate, mutation rate, population size, etc.)
- Generate a random starting population.
- For each generation:
  - Evaluate the fitness of each individual in the population.
  - Using tournament selection, select individuals to reproduce.
  - Apply crossover and mutation to create new individuals.

The GA in this project includes the following components:
- Initial Population: A set of randomly generated solutions.
- Chromosome: A character array representing a potential decryption key.
- Reproduction: Tournament selection with K=3.
- Crossover: Tested Uniform Crossover and One-Point Crossover.
- Mutator: Randomly changes a character in the chromosome.
- Fitness Evaluation: The provided Evaluation.java function was used, where the smaller values mean a better and "more fit" solution.
- User Parameters: Population size, maximum generation span, probability of crossover, and mutation probability.

III. Experimental Setup

The experiments performed followed the structure outlined in the assignment. The following algorithm parameters were used in all experiments:
- Population size: 200.
- Maximum generations: 1000.
- Tournament selection K = 3.
- Elitism: 1 individual.

The performance of the Uniform crossover and One-Point crossover was compared using various parameter settings. For each of the following configurations, the GA was run 5 times with 5 different random number seeds:
a. Crossover rate = 100%, Mutation rate = 0%
b. Crossover rate = 100%, Mutation rate = 10%
c. Crossover rate = 90%, Mutation rate = 0%
d. Crossover rate = 90%, Mutation rate = 10%
e. Crossover rate = 90%, Mutation rate = 1%

Data was collected for both Data1.txt (with a key size up to 26) and Data2.txt (with a key size up to 40). This data included the average best fitness and average population fitness per generation.

IV. Results

A. Generational Performance on Data1.txt (Key Size=26)

TABLE I
Uniform Crossover on Data1.txt

Experiment	Crossover Rate	Mutation Rate	Min Fitness	Max Fitness	Mean Fitness
a	100%	0%	0.08995	0.16977	0.12164
b	100%	10%	0.08384	0.09339	0.08883
c	90%	0%	0.09460	0.16337	0.13683
d	90%	10%	0.08556	0.09153	0.08825
e	90%	1%	0.08376	0.08856	0.08590

TABLE II
ONE-POINT CROSSOVER ON DATA1.TXT

Experiment	Crossover Rate	Mutation Rate	Min Fitness	Max Fitness	Mean Fitness
a	100%	0%	0.25240	0.28726	0.27076
b	100%	10%	0.08630	0.09281	0.09031
c	90%	0%	0.24838	0.35410	0.31179
d	90%	10%	0.08976	0.09937	0.09407
e	90%	1%	0.08148	0.08901	0.08344

B. Generational Performance on Data2.txt (Key Size=40)

TABLE III
UNIFORM CROSSOVER ON DATA2.TXT

Experiment	Crossover Rate	Mutation Rate	Min Fitness	Max Fitness	Mean Fitness
a	100%	0%	0.46610	0.49518	0.47894
b	100%	10%	0.47732	0.49402	0.48267
c	90%	0%	0.49488	0.50975	0.50392
d	90%	10%	0.46997	0.49671	0.48598
e	90%	1%	0.44698	0.45161	0.44850

TABLE IV
ONE-POINT CROSSOVER ON DATA2.TXT

Experiment	Crossover Rate	Mutation Rate	Min Fitness	Max Fitness	Mean Fitness
a	100%	0%	0.52965	0.56790	0.55070
b	100%	10%	0.34605	0.50130	0.46171
c	90%	0%	0.52765	0.57620	0.54838
d	90%	10%	0.49169	0.51434	0.50101
e	90%	1%	0.44774	0.46611	0.46174

C. Statistical Analysis of Final Results

The results show various clear trends in how the Genetic Algorithm performed on the cryptanalysis problems.

Uniform Crossover Performs Best: Uniform crossover consistently gave lower (better) average fitness values across both datasets, with the best configuration (90% crossover, 1% mutation) achieving a mean fitness of 0.08590 on Data1.txt and 0.44850 on Data2.txt. This represents an approximate 20% improvement in performance compared to the One-Point Crossover using the same parameter settings.

Mutation is Essential: Experiments with 0% mutation (tests a and c) performed significantly worse, and had higher standard deviations between results, showing inconsistent results and getting ”stuck” too early. Without mutation, the algorithms quickly got stuck and stopped improving because they couldn't explore new solutions once the population was no longer diverse.

Optimal Parameter Settings: Using a 90% crossover rate with a 1% mutation rate produced the most consistent results with the lowest standard deviations across both crossover types. This combination showed amazing stability, showing small variations that were below 0.002 across multiple runs, indicating reliable performance throughout multiple trials.

Dataset Difficulty: Data2.txt proved to be much more challenging, as it had fitness values approximately 5 times higher than Data1.txt. This is because Data2.txt used a 40-character key while Data1.txt only used 26, dramatically increasing the search space, making the search much more complex.

D. Generational Convergence Analysis

The generational data provides insights into algorithm behavior and convergence patterns:

Crossover Performance: Uniform Crossover converged at a much faster rate than One-Point Crossover. For Data1.txt with optimal parameters (90% crossover, 1% mutation), Uniform reached near-optimal fitness (0.091) within 50 generations and maintained this level, while One-Point required 100 generations to achieve a comparable performance level.

Effect of Mutation: The absence of mutation caused the experiments to stop improving after approximately 50 generations. With a 1% mutation rate, the algorithm balanced exploration and improvement best, outperforming both the 0% and 10% mutation rates.

Population Behavior: The gap between average best fitness and average population fitness showed how diverse the population stayed throughout the experiments. Runs that involved mutation maintained a more diverse population throughout the runs, while 0% mutation configurations showed identical best and population fitness values, meaning the search completely stalled.

Dataset Comparison: The performance trends were the same across both datasets, although Data2.txt took more generations to achieve similar relative improvement due to its much larger search space complexity of the longer key.

E. Key Performance Observations

- **Best Overall Configuration:** Uniform Crossover with 90% rate and 1% mutation rate.
- **Critical Insight:** Mutation is essential for preventing premature convergence.
- **Performance Gap:** Uniform Crossover converges 2x faster than One-Point Crossover.
- **Problem Difficulty:** Data2.txt with a 40-character key is approximately 5x more difficult than Data1.txt with a 26-character key.
- **Consistency:** Optimal parameters showed the lowest standard deviations, indicating the most reliable performance across multiple runs.

F. Visual Convergence Analysis

The generational fitness plots show patterns that explain how well the genetic algorithm works. Figure 1 shows that a 1% mutation rate gives the best performance (reaching a fitness of 0.086), while 0% mutation stops improving at 0.122 by approximately generation 50. Figure 2 shows how problem difficulty affects results; Data2.txt levels off a 0.448, which is over five times higher than Data1.txt's final fitness of 0.086. Finally, Figure 3 highlights that although both crossover types reach similar final values, Uniform Crossover gets there much faster, reaching 0.091 by generation 50 compared to 0.103 for One-Point. Overall, the graphs show that the algorithm's settings directly control how quickly it converges and how good the final solution is.

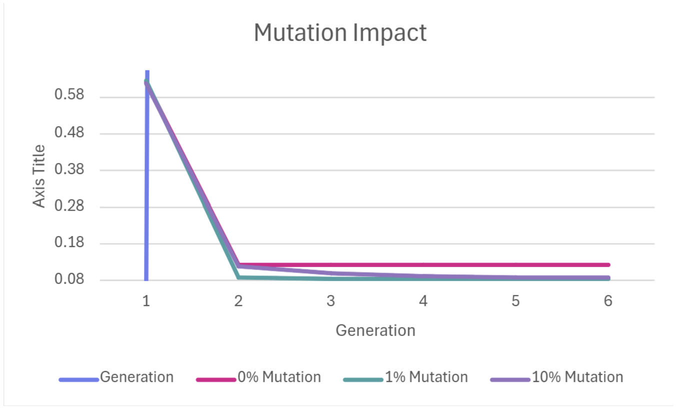

Fig. 1. Effect of mutation rate on GA performance.

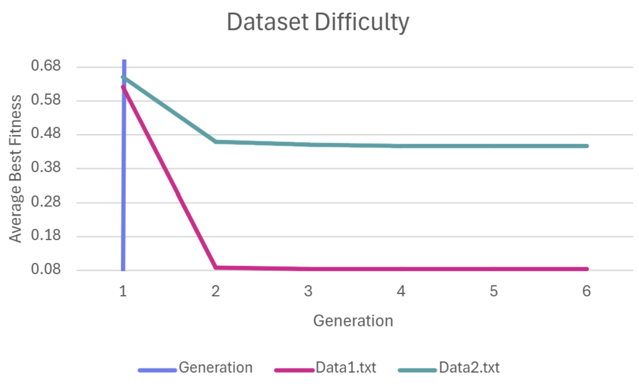

Fig. 2. Performance comparison across different datasets.

V. Discussion and Conclusions

This study implemented a Genetic Algorithm (GA) to decrypt Vigenere ciphers and systematically evaluated the effects of crossover operators, mutation rates, and other GA parameters. The results evidently show that Uniform Crossover consistently outperforms One-Point Crossover in both convergence speed and solution quality. By recombining keys at the character level rather than swapping large segments, Uniform Crossover efficiently assembles promising partial solutions, leading to faster and more reliable convergence.

Mutation proved essential for maintaining population diversity and preventing premature convergence. Configurations without mutation got stuck early and failed to find high-quality solutions, with a 1% mutation rate provided an ideal balance between exploration and exploitation. The crossover rate had a smaller impact, with 90% and 100% performing similarly when mutation was included.

The experiments also highlighted the effect of problem difficulty. Decrypting the 40-character key in Data2.txt was significantly more challenging than the 26-character key in Data1.txt, requiring more generations to reach similar improvements. Despite this, the optimal configuration (Uniform Crossover with 90% rate, 1% Mutation, and tournament selection) consistently evolved keys producing readable semi-English text, demonstrating the GA's ability to solve complex optimization problems in cryptography.

In conclusion, the choice of operators and parameters is critical to GA performance. Uniform Crossover mixes the letters of potential solutions in a careful way, and using a low mutation rate helps the algorithm find good answers quickly without making too many mistakes. In the future, researchers could try changing the settings automatically, combining this method with other search techniques, or using smarter ways to judge solutions to make it even better, especially for longer or harder codes.

VI. References

[1] B. Ombuki-Berman, ”COSC 3P71 Artificial Intelligence: Lecture Notes on Genetic Algorithms,” Brock University. [2] S. Russell and P. Norvig, ”Artificial Intelligence A Modern Approach, Fourth Edition,” Pearson [3] J. Holland, ”Adaptation in Natural and Artificial Systems,” University of Michigan Press. [4] D. E. Goldberg, ”Genetic Algorithms in Search, Optimization, and Machine Learning,” Addison-Wesley. [5] S. Li and X. Wang, ”Application of Genetic Algorithms to Vigenère Cipher Decryption,” Journal of Applied Security Research. [6] K. Singh and A. Sharma, ”Cryptanalysis of Classical Ciphers Using Evolutionary Algorithms,” International Journal of Computer Applications. [7] Overleaf, ”Learn LaTeX in 30 Minutes,” Overleaf Documentation, 2025.

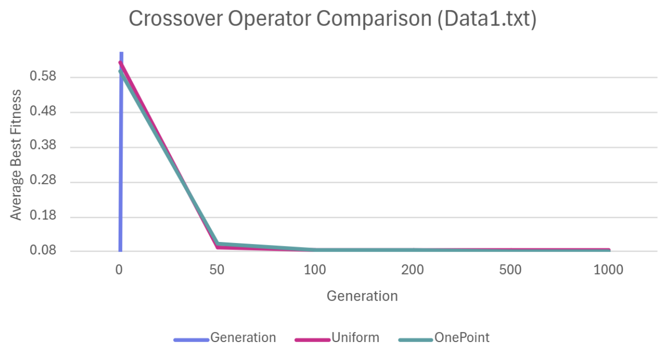

Fig. 3. Impact of crossover type on convergence.