Optimizing a Route in Hollow Knight Game with Genetic Algorithms

Dias dos Santos, Cícero *20221995*; Cordeiro Batista, Francisco *20221847*; Mourão Martins, Lourenço *20222043*; Escarigo Miranda, Vicente *20221845*

Lisbon, Portugal | NOVA Information Management School | 2023-24 Academic Year

Abstract—**The Traveling Salesman Problem (TSP) is a classic optimization challenge that involves finding the shortest possible route that visits a set of locations and returns to the origin. In the context of the game, Hollow Knight, a similar problem arises where the objective is to maximize the amount of GEO, the in-game currency, collected by visiting various locations efficiently. This project explores the application of a Genetic Algorithm (GA) to solve this optimization problem. By simulating the process of natural selection, GAs iteratively improve potential solutions through selection, crossover, and mutation. This implementation aims to determine the most effective route for maximizing Geo collection in Hollow Knight. The results demonstrate the potential of Genetic Algorithms to address complex optimization problems in gaming environments, offering insights into strategic gameplay enhancements.**

Keywords- ***genetic algorithms; traveling salesman problem; optimization problem.***

I. INTRODUCTION

The Travel Salesman Problem is a famous optimization problem based on finding the shortest route between a set of points. Using Genetic Algorithms, TSP is often used in Computer Science to find the best route for data to travel between various nodes. It was first described, in the 1800s, by an Irish mathematician called W. R. Hamilton and a British mathematician called Thomas Kirkman with the creation of a game [1]

Ideally every combination possible would be tested in order to achieve a global optimum, in most cases however this option is unviable due to its large computational expense. As a more efficient alternative, researchers have developed heuristic approaches that provide probabilistic outputs. These approaches significantly reduce computational efforts while delivering optimal or near-optimal solutions. [1]

The Hallow Knight game problem is a variation of the traditional TSP problem, which aims to identify the optimal route. Being a maximization problem, the target was achieving a sequence that yielded the most 'points' possible, or in the project context – 'Geo'. The player must navigate through ten different game areas, having fixed start and end points in Dirtmouth, while adhering to several constraints.

The various constraints placed in the project included: fixed start and end points; forbidden sequences and placement of areas within the route; and specific ratios of Geo earnings and losses between game areas.

The challenge involved creating a pipeline that allows the algorithm to learn the most robust way to maximize Geo earnings while respecting these constraints. The implementation required researching and implementing different genetic operators for crossover, mutation, as well as offspring optimization techniques to improve the Genetic Algorithm's performance.

II. PROPOSED ALGORITHM

The proposed algorithm for solving the optimization problem in "Hollow Knight" is based on Genetic Algorithms (GAs), which is inspired by Charles Darwin's theory of natural selection and the process of evolution. GAs simulate the survival of the fittest among individuals over successive generations for solving optimization and search problems. The algorithm begins with a population of candidate solutions, each represented by a chromosome. Through iterative processes of selection, crossover, and mutation, the algorithm evolves the population. Selection involves choosing the fittest individuals based on a fitness function, which in this case measures the effectiveness of a route in maximizing Geo earnings while adhering to game constraints. Crossover, akin to biological reproduction, combines pairs of chromosomes to produce offspring, potentially inheriting the best traits from each parent. Mutation introduces random changes to individual chromosomes to assure genetic diversity within the population and avoid local optima. By continually selecting, breeding, and mutating individuals, the GA iteratively improves the population converging to an optimal solution for maximizing in-game currency collection.

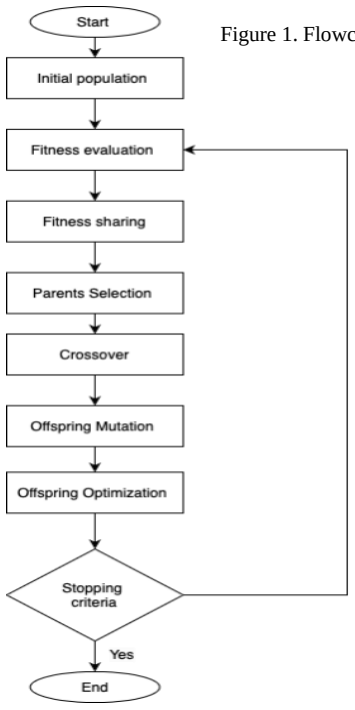

Figure 1. Flowchart of the Genetic Algorithm.

A. *Initial population*

The initial population was generated by randomly permuting genes, each representing a distinct area, in n chromosomes. The only restriction placed during the creation of the first individuals was the fixed start and end points. This approach was designed to ensure a vast amount of diversity within the population.

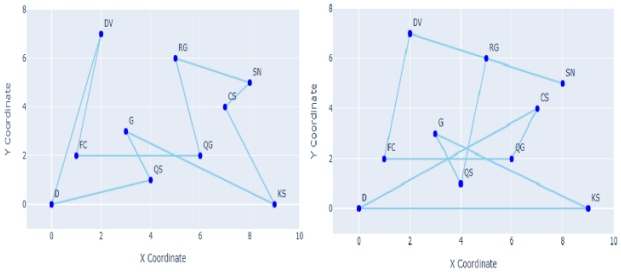

Figure 2. Visual Representations of Individuals.

B. *Fitness Evaluation*

The fitness function in genetic algorithms plays a crucial role in evaluating an individual's adaptability and proximity to the best solution. In this Hallow Knight route optimization, the fitness function primarily calculates the Geo balance of the route. However, the individuals that didn't meet the constraints were penalized, being given a low fitness. This penalization made possible the existence of invalid individuals within the population, which increased diversity, while still being able to converge to optima.

To calculate the Geo balance of each route, a gain/loss matrix was used. Based on the first matrix given in the theoretical project file, a random matrix generator function was developed to produce a similar range of values, maintain the same ratio of negative to positive values, and adhere to the constraint regarding the Geo gain ratio between "*Greenpath*" and "*Forgotten Crossroads*," two areas of the game. The ability to generate different matrices enabled a high degree of robustness for the genetic algorithm. While still being able to feed the algorithm with data that resembled its future input.

C. *Fitness Sharing*

Finding the optimal solution in the Traveling Salesman Problem (TSP) can be challenging due to local optima, which make it difficult to escape and discover better routes. To address this, genotypic and phenotypic diversity were first computed and calculated to observe the diversity of solutions throughout the generations. Consequently, shared fitness was implemented, greatly improving the progression of the algorithm by preventing early convergence.

Fitness sharing operates by penalizing observations that are similar to each other, thus promoting exploration of the solution space. In this case, due to the characteristics of the problem, the distance metric utilized was genotypic diversity. Genotypic diversity refers to the variation in the genetic makeup (genotypes) of individuals within a population. Using this metric, a fitness sharing function was inserted into the pipeline immediately after the initial fitness evaluation, which artificially lowered each individual's fitness. The reduction in fitness was inversely proportional to their genotypic diversity. Partially Mapped Crossover (PMX): PMX functions by Adopting this two-step process of fitness calculation before parent selection significantly facilitated the escape from local optima.[6]

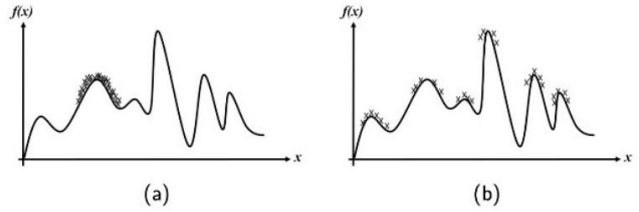

Figure 3. Fitness Sharing Example. Source: [6].

D. *Parents Selection*

In the study of genetic algorithms, selection operators handle determining which individuals from a population will contribute to subsequent generations. The process of choice is vital as it enforces natural selection principles, thereby favoring individuals with higher fitness and promoting the survival of the fittest.

Selection algorithms ensure that the choice of individuals is both probabilistic and fitness based. Probabilistic selection means that repeating the selection process under identical conditions can yield different outcomes, allowing for population diversity. Fitness-based selection ensures that individuals with superior fitness have a higher probability of being chosen compared to those with lower fitness, thereby prioritizing the most adapted individuals. Moreover, every individual, regardless of fitness, has a non-zero probability of choice. This introduces a degree of randomness, reflecting natural scenarios where less adapted individuals may survive due to fortunate events.[2]

For this project, three probabilistic selection algorithms were evaluated: Roulette Wheel, Rank, and Tournament. Roulette Wheel Selection, also known as Fitness Proportionate Selection, offers a fitness-proportional probability for each individual to be selected for reproduction. Rank Selection adopts a similar approach by ranking individuals based on fitness. However, this method does not account for the absolute difference between fitness scores, focusing solely on the position in the ranking. Lastly, Tournament Selection involves selecting a group k of individuals from the population and choosing the fittest individual from this group. This process benefits from providing a greatly adjustable balance between selection pressure and population diversity. The implementation of Tournament Selection randomly chose a value of k between 3 and 6 in each iteration.[2]

E. *Crossover*

A crossover operator is like a matchmaker in genetic algorithms, blending the genetic material of two parents to create new offspring. By swapping bits of genetic information at specific points, known as "crossover points," it mimics the natural process of genetic recombination in sexual reproduction. The result is a new individual that combines traits from both parents. In this project, were implemented 4 different crossover algorithms: Partially Mapped Crossover, Order Crossover, Fast Ordered Mapped Crossover, Cycle Crossover
selecting two crossover points and then partially mapping the

genes between these points from one parent to another. The segments of genes are then exchanged, ensuring that the resulting offspring inherit genetic material from both parents without duplication. PMX is particularly useful for problems where the order of elements matters, such as in optimizing routes or schedules.

Order Crossover (OX): OX involves selecting a subset of genes from one parent and then completing the offspring's genetic sequence by incorporating elements from the second parent while maintaining their order. OX is capable of generating diverse offspring while preserving the sequential integrity of genetic elements, which was fundamental in the implementation.

Fast Ordered Mapped Crossover (FOMX): Fast Ordered Mapped Crossover (FOMX) is an optimized version of the traditional genetic crossover operator aimed at improving computational efficiency. By integrating principles from ordered crossover with innovative mapping techniques, FOMX accelerates the generation of offspring while minimizing computational overhead.[4]

Cycle Crossover (CX) operates by identifying cycles of genes within parent chromosomes and then exchanging these cycles to create diverse offspring. CX ensures the preservation of genetic diversity and facilitates exploration of different solution spaces. It is commonly used in permutation-based optimization problems where maintaining diversity and exploring various solutions are essential for achieving optimal results.

F. Offspring Mutation

The mutation operators are fundamental to ensure genetic diversity in the population as well as the progression of generations. They oversee the search space for exploring and avoiding local optima to reach a global optimum solution. Mutation operators cause small random changes in the individuals, which could result in new genetic combinations that crossover alone might not achieve. Three distinct mutation operators were tested: Swap Mutation, Inversion Mutation, and Displacement Mutation.[3].

Swap Mutation works by selecting two random positions in the individual's representation and swapping their values. Although it is not a sophisticated method, it can be particularly useful in permutation-based problems, such as the Traveling Salesman Problem (TSP), where the order of elements matters.

Inversion Mutation is a type of mutation that reverses a randomly selected contiguous segment of an individual's chromosome. This type of mutation enables changes in the genetic makeup of the individual without disturbing the integrity of the sequence. By inverting a segment, the mutation provides a new perspective on the arrangement of genes, which leads to new combinations that may result in better solutions.

Displacement Mutation is more disrupting compared to swap and inversion mutations. It selects a contiguous segment of the individual's chromosome, removes it, and reinserts it at a different position. This mutation can greatly alter the structure of the individual; hence, it is a powerful tool for getting out of local optima and exploring diversified areas of the search space. The ability to move large segments of genes with the displacement mutation allows the algorithm to find new configurations.[3]

G. Offspring Optimization

The proposed algorithm incorporates 2-Opt Heuristic method to optimize offspring routes. It works by swapping genes and sequentially checking the updated fitness value of the chromosome. In case the updated fitness value is better, the gene manipulation is validated, and the offspring suffers that change. Offspring optimization can significantly enhance the quality of solutions by refining the routes generated.

H. Stopping Criteria

The iteration terminates upon reaching a pre-determined number of generations.

III. RESULTS AND DISCUSSION

A. Hyperparameter Optimization

To gain a deeper understanding of the impact of n, the number of iterations associated with 2-Opt, a comprehensive analysis was conducted using a fixed set of parameters. By generating Geo matrices with random seed numbers, each value of n was tested in 100 runs of the genetic algorithm, ensuring that all other parameters remained constant. Results showed that the value of 5 provided the highest average best fitness. Although 2-Opt revealed a considerable fitness boost compared to not using 2-Opt at all, most other values did not show as significant a boost. For values larger than 40, fitness values were comparable to not using offspring optimization. In terms of diversity, 2-Opt presents a tradeoff between phenotypic and genotypic diversity. While phenotypic diversity decreases due to the convergence of solutions to similar structures, genotypic diversity increases as the genetic algorithm explores different combinations and sequences of elements. Regarding average total running time, it is represented by a positive linear function, as expected.

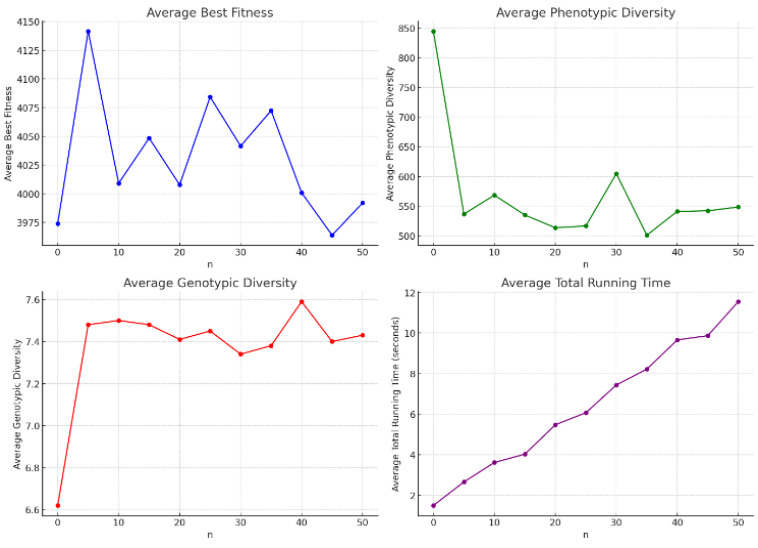

Figure 4. Changes in Algorithm's Performance depending on the n parameter in two-opt.

Additionally, the impact of using elitism in the genetic algorithm was tested. A grid search was utilized once again to test various combinations of parameters for the genetic algorithm. The configuration of the grid search involved 40 iterations for each combination of hyperparameters. Different setups of operators were utilized with and without elitism to understand how it affected the genetic algorithm's convergence to solutions. To conduct this analysis, the grid search was set up as follows.

TABLE II
HYPERPARAMETER GRID FOR ELITISM ANALYSIS

Hyperparameter	Values
Population Size	50
Num Generations	50
Mutation Rate	0.05
Crossover Rate	0.7
Elitism Size	2
Selection	tournament_selection, roulette_selection, rank_selection
Crossover	partially_mapped_crossover, fast_order_mapped_crossover, order_crossover, cycle_crossover
Mutation	swap_mutation

The grid search's results suggested that the incorporation of elitism had a positive influence on the genetic algorithm's performance. To facilitate visualization, the data was organized into two separate tables, with each table aggregating combinations based on the type of crossover and selection algorithm used.

TABLE III
PERFORMANCE OF DIFFERENT CROSSOVER FUNCTIONS

Crossover Operator	Avg. Best Fitness with Elitism	Avg. Best Fitness without Elitism
partially_mapped_crossover	4112.99	3895.36
order_crossover	4079.49	3924.86
cycle_crossover	4018.93	3964.90
fast_order_mapped_crossover	3991.23	3928.26

TABLE IV
PERFORMANCE OF DIFFERENT SELECTION ALGORITHMS

Selection Function	Avg. Best Fitness with Elitism	Avg. Best Fitness without Elitism
roulette_selection	3979.58	3814.05
tournament_selection	3976.59	3924.32
rank_selection	4023.84	3998.28

After the conclusion of the genetic algorithm pipeline and the implementation of several operators, the algorithm was subject to various combinations of hyperparameters to determine the ideal settings. The relatively small complexity of the problem and moderate size of the hyperparameter grid allowed for the use of an exhaustive optimization method. Therefore, it was opted for a grid search with a total of 15 iterations. For each iteration a Geo matrix was generated utilizing a seed number and each possible combination within the hyperparameter grid was tested. After the 15 iterations concluded, an average fitness score dictated the optimal settings.

TABLE V
HYPERPARAMETER GRID FOR GENETIC ALGORITHM OPTIMIZATION

Hyperparameter	Values
Initializer	population
Evaluator	fitness_function
Population Size	50, 100
Num Generations	50, 100
Mutation Rate	0.05, 0.1
Crossover Rate	0.7, 0.9
Elitism Size	2, 5
Selection	tournament_selection, roulette_selection, rank_selection
Crossover	partially_mapped_crossover, fast_order_mapped_crossover, order_crossover, cycle_crossover
Mutation	swap_mutation, displacement_mutation, inversion_mutation

TABLE VII
IDEAL PARAMETERS OF THE GRID SEARCH

Parameter	Value
Initializer	population
Evaluator	fitness_function
Population Size	100
Num Generations	50
Crossover Rate	0.7
Elitism Size	2
Selection	rank_selection
Crossover	partially_mapped_crossover
Mutation	swap_mutation

IV. CONCLUSION

This project successfully implemented a genetic algorithm to address a complex optimization problem similar to the Traveling Salesman Problem (TSP). Utilizing genetic operators such as selection, crossover, and mutation, along with techniques like the 2-Opt Heuristic for offspring optimization and fitness sharing functions, the project achieved a robust and efficient solution. These methods facilitated the evolution of solutions across generations, enhancing route quality and avoiding local optima.

Hyperparameter optimization was crucial in refining the genetic algorithm's performance. Various genetic operators and strategies were tested to identify the optimal settings, significantly impacting the algorithm's convergence and efficiency.

In conclusion, this research demonstrates the adaptability and effectiveness of evolutionary algorithms in solving complex optimization problems. The findings provide a solid foundation for strategic improvements in various fields requiring intricate resource management and route optimization.

REFERENCES

[1] TechTarget. (n.d.). Traveling salesman problem (TSP). in WhatIs.com. Retrieved from https://www.techtarget.com/whatis/definition/traveling-salesman-problem

[2] Vanneschi, L., & Silva, S. (2023). Lectures on Intelligent Systems. In Springer eBooks. https://doi.org/10.1007/978-3-031-17922-8

[3] TutorialsPoint. (n.d.). Genetic Algorithms - Mutation. in TutorialsPoint. Retrieved from https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_mutation.htm

[4] Lakshmi, R., & Vivekanandan, K. (2015). "Performance analysis of a novel crossover technique on permutation encoded genetic algorithms." IEEE Xplore. https://ieeexplore.ieee.org/document/7105281

[5] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019.

[6] Vanneschi, L., "Genetic Algorithm" from theoretical slides