ECE 470 - Artificial Intelligence

Genetic Algorithm for Golf Putt Optimization

Team 18- Green Demons

Report Submitted on: 3 August 2025

Names:

1. Anitta Varghese (V00984911)
2. Mattias Kroeze (V00934043)
3. Mudit Jaswal (V00982906)

Table of Contents

1. Summary

This project explores the use of Genetic Algorithms (GAs) to optimize golf putting strategies in a simulated physics environment. By modeling the problem as a continuous control task in NVIDIA Isaac Sim, our system determines an optimal velocity vector (X, Y, Z) that "sinks" a golf ball from randomized positions on procedurally generated greens. The project demonstrates the GA's ability to find viable putting strategies across varying terrains and distances. Early stopping and velocity-sensitive fitness functions ensure realistic solutions that don't overshoot the hole. Our results show strong convergence toward successful shots, especially with a refined reward design and parameter tuning.

2. Introduction

2.1. Background

In recent years, artificial intelligence has seen remarkable success across various fields, including robotics and strategic game environments. Among these techniques, Genetic Algorithms have emerged as a powerful optimization tool inspired by the process of natural selection. Golf putting is an accuracy-focused task that demands careful consideration of terrain topography, distance, and direction. Simulating and computing solutions for putting strategy presents an interesting challenge that merges physics with decision making. Artificial intelligence, in the form of neural networks and evolutionary algorithms, has been extensively applied to problems of this sort and provides adaptive solutions without requiring explicit rules.

Our project utilizes IsaacLab, a powerful physics-based platform from NVIDIA, to create a simulated environment in which to train an AI putting agent. The issue is framed as a continuous control problem, for any spot on a procedurally generated putting green, the agent must determine the optimal strength and velocity with which to sink the putt.

2.2. Motivation

Green-reading is one of the most challenging aspects of golf, even for experienced players. A visual tool that models putting strategy and simulates ideal ball paths can be valuable for both coaching and analysis. Our motivation is to develop a system that visualizes optimal putting lines based on the terrain and the initial ball position.

Advancements in simulation platforms such as IssacLab provide high fidelity environments to replicate realistic putting greens with diverse terrain features. This enables experimentation and optimization that would be difficult or costly to achieve in real world environment. By formulating putting as a continuous optimization problem, finding the best velocity vector to sink the ball, this project aims to demonstrate how genetic algorithm can be used in complex, physics- based scenarios. Overall, this work seeks to push the boundaries of AI' s ability to solve skill-based tasks, while also providing a realistic demonstration on how intelligent agents can adapt to various physical environments.

3. Related Work

A relevant study by A.Coskun, "Optimization of Mini-Golf Game using the Genetic Algorithm" [2], uses a simple 3D mini-golf simulator to train a robot to strike a ball towards a goal point, while avoiding random obstacles. Each action was encoded as a chromosome representing the x and y components of a force vector and the robot is rewarded based on the closeness of ball to the target. Coskun's results demonstrated that even a simple genetic algorithm can work efficiently under obstacle-filled conditions. However, the approach focused on discrete actions and fixed obstacle placements, with minimal attention to terrain topography.

The genetic algorithm used in Coskun's approach employed standard evolutionary techniques including selection, crossover, and mutation. The study demonstrated how parameter choices such as population size, elitism, and mutation rates influenced convergence time and solution quality. While the system successfully navigated simple obstacle arrangements, it did not incorporate factors such as generalization across various green profiles or multi-shot strategies.

Our project builds on this by introducing more of a continuous action space and a more realistic physics simulation. Our simulation emphasizes green-reading and precise ball control across varied velocities, meanwhile Coskun's work is focused on a static environment. Other applications of GAs in similar context, such as path planning and physics-based control, reinforce the viability of the approach.

4. Problem Formulation

We treat each putt as a continuous single-action search problem. The goal is to find a 3-dimensional initial velocity vector that sinks the ball with a realistic entry speed. During each generation, we apply the velocity vector to the ball once and observe the ball as it rolls for 500 timesteps, equivalent to 5 real-world seconds. A ball is considered “sunk” when its center comes within the radius of the cup (54 mm) and is traveling below 10 m/s. However, in real-world play, a velocity between 1–2 m/s is optimal to ensure the ball is sunk with control and does not skip out. We intentionally set the maximum allowed velocity to be quite high so the genetic algorithm can find a valid solution first, then gradually refine the shot to achieve a controlled sink.

To guide this search, we implement a two-phase fitness function. If the ball is not successfully sunk, we award up to 1000 points based on the ball’s final distance at the end of the 5-second episode, where a distance equal to the hole radius gives the full score and reward drops off quickly as distance increases. Once the ball is sunk, we reward based on the reciprocal of the entry velocity. A velocity of 10 m/s (our upper limit) earns 1000 points, while a velocity of 0.5 m/s earns up to 20,000. Any solution that results in a sink under 1 m/s is considered optimal and causes the evolution to terminate immediately. Additionally, if performance stagnates for ten consecutive generations, we stop early, as further improvement is unlikely.

5. Methodology and Evaluation

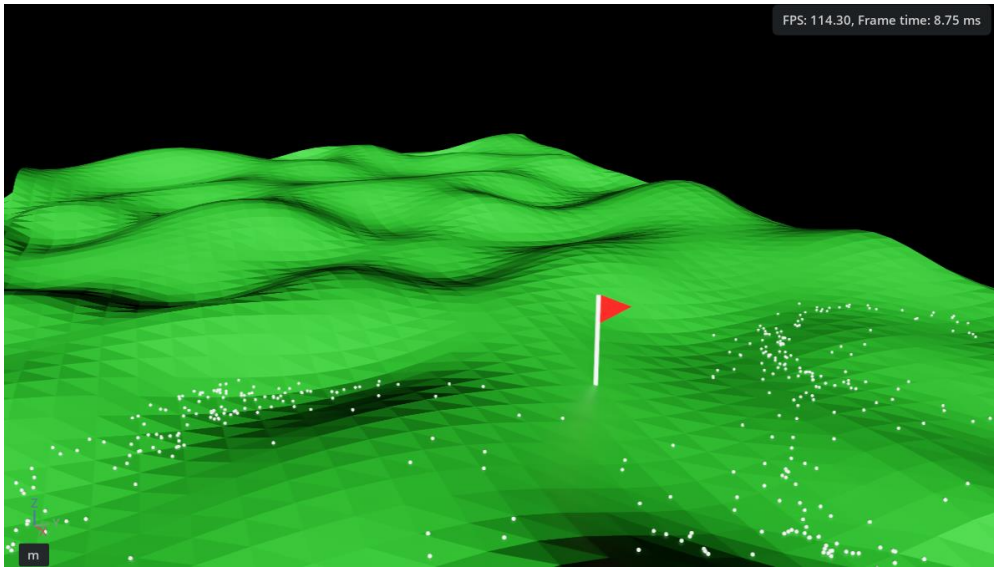

Figure 1: Screenshot of simulation environment, showcasing procedurally generated green and parallel ball environments

To evaluate the performance of velocity vectors, we created an environment in NVIDIA's Isaac Lab, a physics simulation platform designed for robotics and machine learning. Given the complexity of simulating physical environments and the need for precise control, the tools offered by Isaac Lab were well suited for this project. To create diverse and challenging greens, each run uses a procedurally generated heightfield terrain with tuned noise ranges and step sizes to produce smooth, rolling hills. After generating the terrain, we select a random distance to the hole and spawn the ball 2 cm above the z-height of the green at that location. The ball is then duplicated across 2048 distinct and parallel environments.

We then generate a uniform distribution of 2048 initial velocity vector genomes as our starting population. Each velocity is applied once, and we simulate the resulting ball path for 500 timesteps. After the episode, each shot is scored using the two-phase fitness function described earlier. We keep the top 5% of performers unchanged as an elite group. A parent pool is formed from the top half of the remaining population. Children are generated using a weighted average of parent genes, which suits the continuous nature of this problem better than binary crossover. Ten percent of child genes are then mutated using Gaussian noise.

Metrics such as best fitness, average fitness, entry velocity, and closest approach are logged each generation to both CSV and TensorBoard. Code snapshots are also saved to ensure reproducibility. A run terminates if a genome earns a score of at least 10,000, if no improvement occurs for ten generations, or if 25 generations are reached without success.

6. Results and Discussions

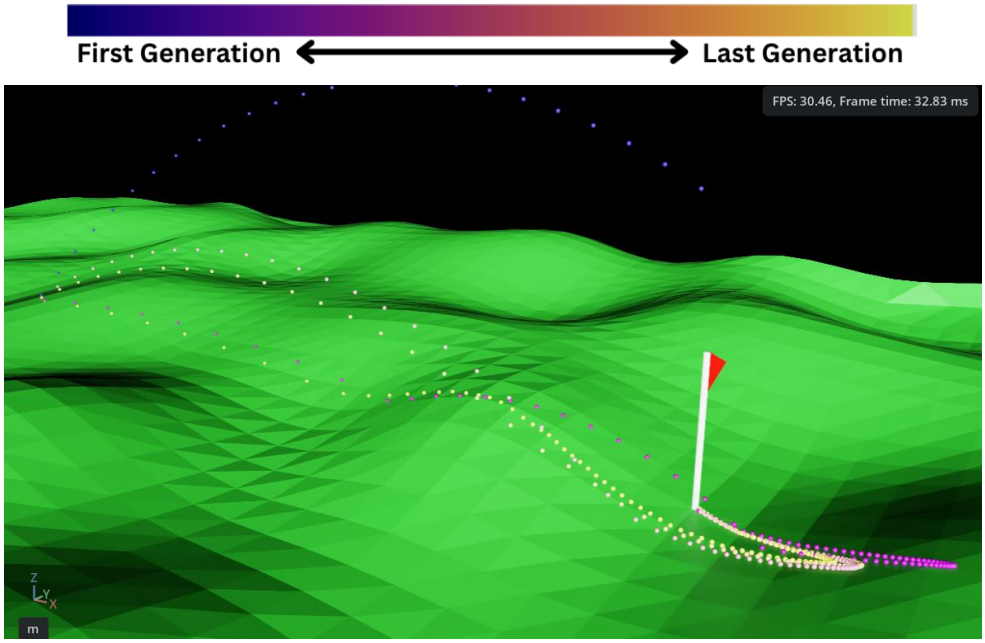

Figure 2: Screenshot of simulation environment, showcasing a trail behind each "breakthrough" to visualize how the optimal shot is refined over generations.

Figure 2 presents a screenshot of the simulation environment during one of these runs. The coloured trails behind each ball represents each time a new best performing genome is found across all the generations. These trails help visualize how the GA gradually refines its search around successful shots.

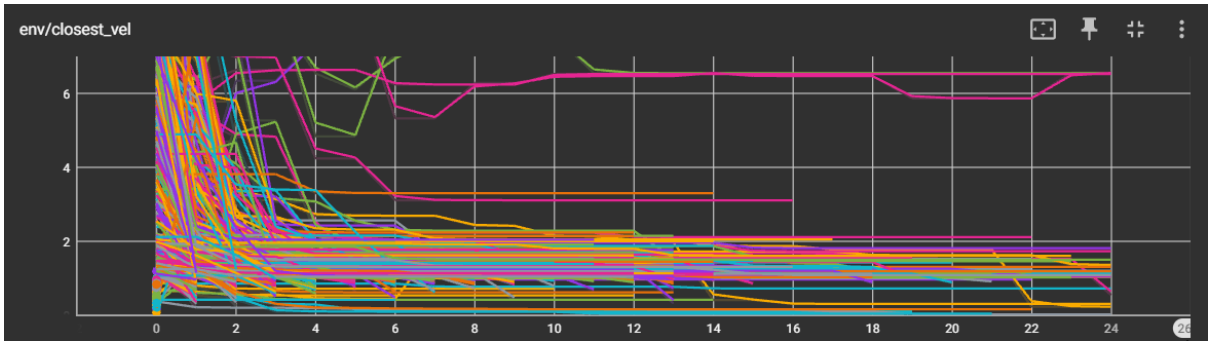

Figure 3: Entry velocity of the generation's best performer, across 800 runs.

Our experiments show that a genetic algorithm (GA) can effectively converge on successful golf putts across procedurally generated terrains. As shown in figure 3, in nearly all 800 runs, the algorithm discovered a valid solution that sinks the ball with a controlled entry speed within 5 to 15 generations.

To get the performance that we were able to achieve from our GA search implementation, there was a significant amount of work went into determining the most effective way to formulate the fitness function, in a way that not only incentivises the ball the reaching the hole, but also rewards the ball being sunk with a low, controlled velocity.

One of the challenges we faced during this project was our initial attempt to use NEAT (NeuroEvolution of Augmenting Technologies) as the learning algorithm. While NEAT is powerful and capable of evolving both the topology and the weights of a neural network, it ended up being an imperfect fit for our setup. The main issue was integrating with Isaac Sim, which added unexpected complexity.

Eventually, we decided to pivot to a traditional Genetic Algorithm which gave us more control and interpretability. Instead of evolving neural networks, we evolved simple 3D velocity vectors directly. Using GA, we were able to successfully generate putts under highly challenging conditions, including difficult terrain and long distances. These scenarios tested the adaptability of our approach, and the algorithm consistently produced viable solutions.

Another challenge was getting the ball to roll smoothly across the procedurally generated greens, there was significant troubleshooting needed to diagnose seemingly random collisions to invisible edges, the solution ended up being reducing the resolution of the terrains mesh to reduce the number of physics API collision checks that needed to be made between the ball and the terrain.

7. Conclusion

This project set out to explore whether a Genetic Algorithm (GA) could effectively optimize golf putts in a simulated, physics-based environment. Through extensive experimentation in Isaac Sim, we demonstrated that even with a relatively simple GA structure and a continuous 3D action space, the algorithm was capable of reliably converging on successful putting strategies across a variety of starting distances and green configurations.

One of the major takeaways from this project is that the simplicity of GAs can be a strength in high-dimensional, physics-based tasks. While we initially explored more complex methods like NEAT, the GA outperformed them in terms of speed, stability, and ease of integration with the simulator. Additionally, the project highlighted the importance of clear reward shaping and thoughtful logging, both of which were critical in tracking convergence and identifying problem areas.

8. Future Work

While our genetic algorithm successfully optimizes putting across diverse terrains, there are factors that could be considered for further enhancement of the project. Future model can integrate variable factors like wind effects, grass friction to better reflect real-life conditions. Moreover, developing visualization tools can help golfers to understand the strategies that can help them putt successfully.

Implementing the system as an online adaptive agent that can adapt to putting strategy in response to real time feedback could potentially enable improved putting performance. Verification of simulation validity of accuracy from testing the optimal putt parameters on a real robotic putting platform would be valuable as it helps to understand the practical experience from real world noise.

We also see potential in exploring sim-to-real transfer methods, such as domain randomization, to bridge the gap between our simulated environment and physical implementations. Lastly, comparing our GA approach with alternative optimization techniques like a refined NEAT setup could help validate our methodology and uncover more efficient solutions.

9. References

[1]ScottyX, “Putting Problems: An Examination,” *GolfDom*, Feb. 28, 2022. https://www.golfdom.com.au/putting-problems-an-examination

[2]A. Coskun, “Optimization of a Mini-Golf Game using the Genetic Algorithm,” Nov. 30, 2011. https://eejournal.ktu.lt/index.php/elt/article/view/180/137

[3] NVIDIA, “Overview-Isaac Lab Documentation,” Github.io, 2023. https://isaac-sim.github.io/IsaacLab/main/index.html