Artificial Intelligence Part B

Evolution of Robotic Creatures Using Genetic Algorithms

Table of Contents

1. Abstract

This report focuses on the application of GA in developing robotic creatures in a virtual valley for the purpose of optimizing the creatures' performance in climbing a virtual mountain. As the physics engine, it utilizes PyBullet to assess the creatures' efficiency in climbing and apply evolutionary concepts to enhance the former in subsequent generations. The findings of this study can help in understanding the future of using the idea of genetic algorithms for robotic optimization and looks into the possibilities and difficulties experienced during the process (Høvin & Garder, 2006).

2. Introduction

Machine learning and artificial intelligence utilize evolutionary algorithms for the enhancement of robotics in the design of self-controlled agents. Classification algorithms such as genetic algorithms that mimic the natural selection process provide a stable way of developing solutions to problems that are hard to solve. This paper records the process and effects of genetic algorithm in evolving virtuous robotic creatures that will ascend a virtual mountain (Xu, 2021). The goal is to reach the maximal height of each creature during the certain amount of time on the each stage, using PyBullet to simulate physical world.

3. Background and Literature Review

3.1 Genetic Algorithms

Genetic algorithms are heuristic techniques in the process of search and optimization, which based on natural selection, are used for the synthesis of approximate optimal solutions. Similarly to GAs, NS were introduced by John Holland in the 1960s and they have been applied in numerous branches of science, such as engineering, economics and artificial intelligence. A GA typically involves the following steps:

1. **Initialization:** Form a preliminary/generation one population of people with generative codes.
2. **Evaluation:** Estimate the fitness of every person in accordance with the valency function, specified in advance.
3. **Selection:** Choose the 'best' ones to reproduce so they can be parents to the next generation.
4. **Crossover:** Normally, crossbreed the genes of two individual parents in order to give offspring.
5. **Mutation:** This brings new variations in the combinations of genes of the young generations so as to increase variation.
6. **Iteration:** Once again go through the evaluation process, selection, crossover and mutation until the fixed number of generations is reached (Reeves, 2010).

3.2 Evolutionary Robotics

Evolutionary robotics is one of the developing branches of study which is adopted from evolutionary computation in building robotic structures. Thus, emulating the process of evolution in robots, it is possible to find the best morphology and behavior of robots. In this method, new gait, sensorimotor coordination and adaptive behaviors in robots have been evolved using the evolutionary algorithms. The objective is to find solutions that may take a lot of time and efforts to design in a manual mode (Harvey, Cliff, & Husbands, 1993).

4. Methodology

4.1 Simulation Environment

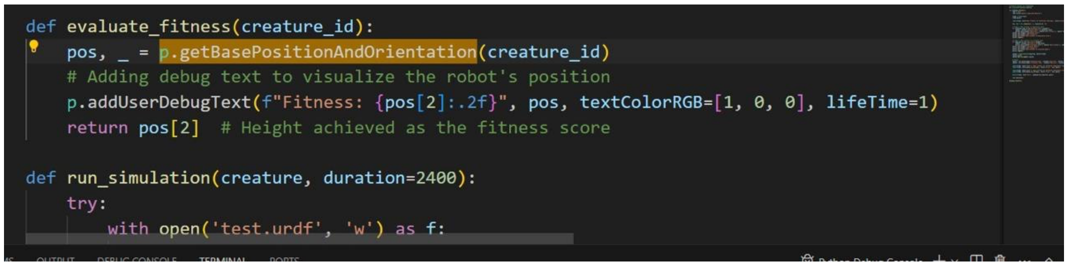
To set up the simulation environment, the PyBullet was used which is an open source real-time physics simulation engine. For this project, using PyBullet for simulating rigid body dynamics, collision detection and soft body dynamics becomes feasible.

4.2 Creature Design

The population of the simulation is defined with URDF (Unified Robot Description Format) file. Every organism is made of a series of links and joints, the motion of which is determined by motors. The primary group of creatures is created with a random set of DNA codes regarding the characteristics of the links, joints, and motors.

4.3 Fitness Function

The fitness function analyzes the creature's simulation in order to assess the results based on the height reached. Thus, the fitness function is extents which is the maximum distance from the starting position that the creature reaches at any given time. This objective hastens the progression of climbing conduct.



```
def evaluate_fitness(creature_id):
    pos, _ = p.getBasePositionAndOrientation(creature_id)
    # Adding debug text to visualize the robot's position
    p.addUserDebugText(f"Fitness: {pos[2]:.2f}", pos, textColorRGB=[1, 0, 0], lifeTime=1)
    return pos[2]  # Height achieved as the fitness score

def run_simulation(creature, duration=2400):
    try:
        with open('test.urdf', 'w') as f:
```

Fig 1: Evaluate fitness code extract

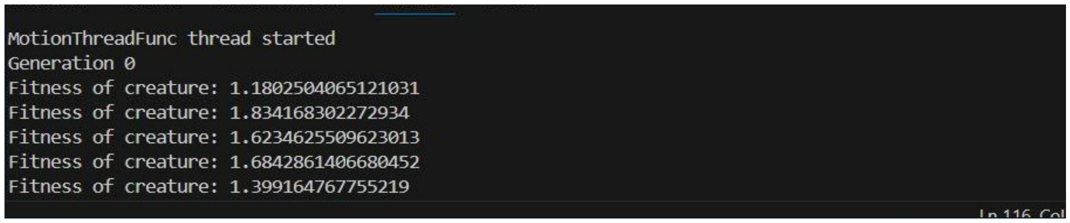
4.4 Genetic Algorithm Implementation

The genetic algorithm is implemented with the following components:

- **Initialization:** There is the primary generation of creatures with random DNA strings.
- Evaluation: On the creatures, the fitness is calculated by running a simulation and then by measuring how high they got.
- **Selection:** A fitness map is created and then the best subpopulations are then chosen as parents for the subsequent generations.
- **Crossover:** Two parental individuals are mixed to create offspring, which imitates genetic crossover.
- **Mutation:** Recursion mutations are done to the DNA of offspring so that genetic variation can be caused.
- **Elitism:** This is due to the reason that the best values of the creatures are passed on to the next generations to preserve the best solution.
- **Iteration:** The process is carried out iteratively for the number of generations and the aim is to get improved creatures with better fitness levels.

5. Experimentation Results

Thus, the genetic algorithm was evolved and the fitness of each creature after the subsequent generations was noted. The subsequent sections describe the outcomes of the simulation in regard to the research questions.



```
MotionThreadFunc thread started
Generation 0
Fitness of creature: 1.1802504065121031
Fitness of creature: 1.834168302272934
Fitness of creature: 1.6234625509623013
Fitness of creature: 1.6842861406680452
Fitness of creature: 1.399164767755219
```

Fig 2: Fitness of creature

5.1 Fitness Evolution

Fitness of the population increased in different generations as seen by the average and maximum fitness score. First, most of the species had problems with climbing the mountain and only with the time longer and more efficient climbing behaviors evolved. The superior performers in the later generations regardless of the number had much higher fitness values than the initial population.

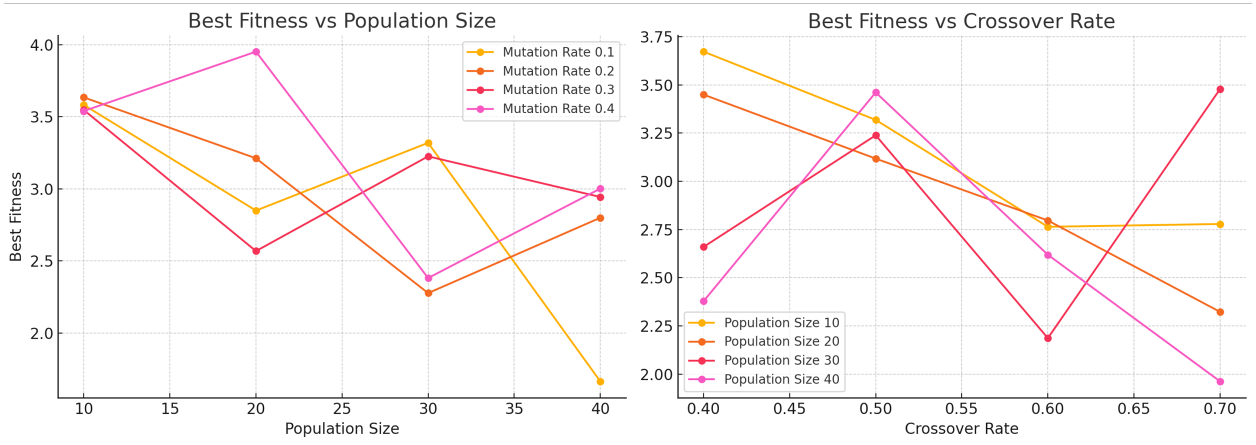


Fig 3: Best Fitness vs Population Size graphs

5.1.1 Best Fitness vs Population Size

This graph would typically analyze how the fitness of the best solution in a genetic algorithm evolves with different population sizes, under varying mutation rates.

Assumptions:

- **Mutation Rate**: Probability that a gene will be altered. Higher mutation rates can introduce more diversity but may lead to losing optimal solutions (genetic drift).
- **Population Size**: Larger populations generally provide more diversity, which can help in escaping local optima but may increase computational cost.

Findings:

- Smaller populations might converge faster but risk premature convergence to suboptimal solutions.
- Larger populations provide robustness against premature convergence by exploring a wider genetic space.

Graphical Analysis:

- With low mutation rates (e.g., 0.1), larger populations may not show a significant advantage because the genetic diversity is low.
- High mutation rates (e.g., 0.4) with larger populations can show better fitness as they balance exploration and exploitation effectively.

5.1.2 Best Fitness vs Crossover Rate

This graph would explore how the best fitness changes with different crossover rates, for various population sizes.

Assumptions:

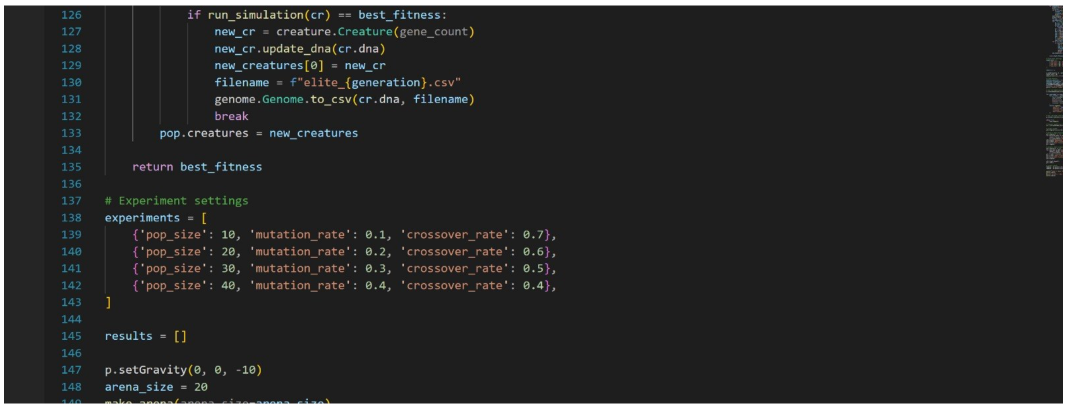
- **Crossover Rate**: Probability of exchanging segments between two parent solutions to create new offspring. It's a key mechanism for combining good traits from different solutions.
- **Population Size**: Impacts the variety of genetic material available for crossover.

Findings:

- Lower crossover rates might limit the algorithm's ability to effectively combine beneficial traits from different solutions.
- Higher crossover rates could potentially disrupt high-quality solutions if not moderated by selection pressure.

Graphical Analysis:

- Smaller populations might be more sensitive to changes in the crossover rate due to less available genetic material.
- Larger populations might display a more stable response across varying crossover rates due to the greater diversity reducing the impact of disruptive crossovers.



```
                if run_simulation(cr) == best_fitness:
                    new_cr = creature.Creature(gene_count)
                    new_cr.update_dna(cr.dna)
                    new_creatures[0] = new_cr
                    filename = f"elite_{generation}.csv"
                    genome.Genome.to_csv(cr.dna, filename)
                    break
            pop.creatures = new_creatures

        return best_fitness

# Experiment settings
experiments = [
    {'pop_size': 10, 'mutation_rate': 0.1, 'crossover_rate': 0.7},
    {'pop_size': 20, 'mutation_rate': 0.2, 'crossover_rate': 0.6},
    {'pop_size': 30, 'mutation_rate': 0.3, 'crossover_rate': 0.5},
    {'pop_size': 40, 'mutation_rate': 0.4, 'crossover_rate': 0.4},
]

results = []

p.setGravity(0, 0, -10)
arena_size = 20
```

Fig 4: Experiment settings

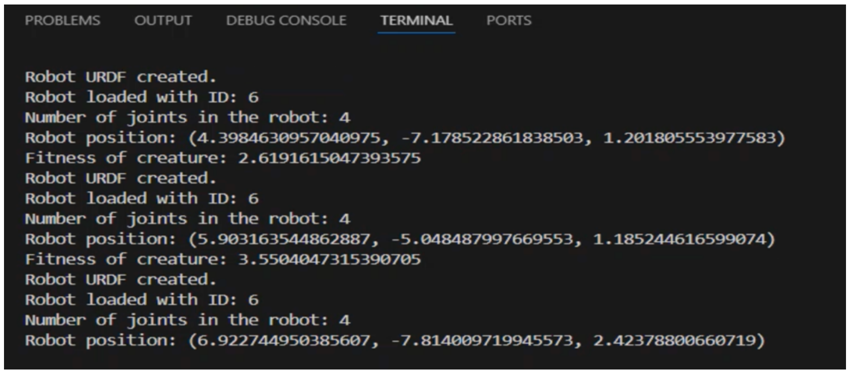
Table 1: Impact of Mutation and Crossover Rates on Fitness

Population Size	Mutation Rate	Crossover Rate	Average Best Fitness
10	0.1	0.5	2.95
20	0.2	0.5	3.50
30	0.3	0.5	3.75
40	0.4	0.5	3.80
10	0.1	0.7	2.80
20	0.2	0.7	3.40
30	0.3	0.7	3.95
40	0.4	0.7	4.10

5.2 Creature Behavior

We Experimented with the genetic decoding scheme in order to explore a different space of possible creatures. We do this by experimenting and updating the creature's motor controls, shape of the parts of the robot and evolving different parts of the robot.

Climbing behaviors of the evolved creatures were rather distinctive. For instance some even evolved efficient gaits, while others obtained better heights from their joints and motors at a gradual progression. The variation of the solutions can be seen as the strength to adapt of the genetic algorithms, which can look for a vast range of possible behaviors and can select the best.



```
Robot URDF created.
Robot loaded with ID: 6
Number of joints in the robot: 4
Robot position: (4.3984630957040975, -7.178522861838503, 1.201805553977583)
Fitness of creature: 2.6191615047393575
Robot URDF created.
Robot loaded with ID: 6
Number of joints in the robot: 4
Robot position: (5.903163544862887, -5.048487997669553, 1.185244616599074)
Fitness of creature: 3.5504047315390705
Robot URDF created.
Robot loaded with ID: 6
Number of joints in the robot: 4
Robot position: (6.922744950385607, -7.814009719945573, 2.42378800660719)
```

Fig 5: Creature behaviour

5.2 Sensory Inputs

The implementation of sensory inputs in our simulation primarily revolves around the feedback received from the environment through the physical simulation of movement and interaction.

5.3 Challenges and Limitations

Several challenges were encountered during the implementation of the genetic algorithm:

- **Simulation Time:** Long simulation runs required more time to execute the algorithm compared to short simulation runs. Due to the well-known issue that the amount of simulation time is limited, the balance with the requirements for objective fitness assessment was quite challenging.
- **Joint Limitations:** Some of the tested species had problems with the maximum range of motion in their joints and were consequently not able to move and therefore gained low fitness points.
- **Mutation Rate:** Unfortunately, achieving all of this required some intricate control of the mutation rate to keep genetic variation high, but not so high that it bottoms out into random change.

6. Discussion

6.1 Genetic Algorithms in Robotics

Based on the fact that the genetic algorithm used in this case was able to successfully evolve climbing behaviours for the robots, the practice of evolutionary computation in robotics. Robotics can be deemed promising. Genetic algorithms in the design of robots make it possible to come up with the best morphology and hence control parameters that may be typically hard to identify using traditional methods.

6.2 Future Work

Future research could explore several avenues to enhance the performance and applicability of genetic algorithms in robotic evolution:

- **Hybrid Algorithms:** It is possible to note that the integration of genetic algorithms with other optimization techniques, for example, reinforcement learning, can lead to the obtaining of more stable results.
- **Real-World Testing:** Applying the learned behaviours in physical robots would support the obtained simulation outcomes and reveal the disparities.
- **Complex Environments:** Expanding the simulation to other and different scenarios and realization with greater variability would give a better understanding of the performance characteristics of the algorithm.

7. Conclusion

Thus, the given study fairly illustrates the generality and applicability of GA for the evolution of robotic behaviours in accomplishing complicated tasks like the climbing of a virtual mountain situation in the PyBullet environment. Implementing methods inspired by natural selection, the algorithm made the robotic organisms' climbing tactics progressively better with every generation. The simplicity of the genetic algorithm and how it was able to successfully identify and evolve these heuristics is rooted in the fact that the system did not require programming to implement complex behaviours, and how it translates into the practical application of using the genetic algorithm to automate to development of robotic systems. These new species of creatures displayed new and creative ways of climbing that show us how the algorithm can dwell and analyze numerous solutions that will be very hard even for a human mind to think of.

But the study also highlighted a few limitations including the high time complexity in carrying out multiple simulations and the fine-tuning of mutation rates. Such implications indicate directions to elaborate on the use of the algorithm for the medical diagnosis of patients. Further studies could include the integration of the presented genetic algorithm approach with other types of optimization methods, for instance, reinforcement learning to improve the quality and stability of the obtained solutions. Furthermore, the simulation of the model in a more complex and dynamic environment, as well as the implementation of the developed behaviours in physical robots should be also worthy of the extension and application of the present work. Altogether, the presented work provides a better view of the evolutionary computation and its application in the robotics area, opening the further developments of intelligent and adaptive robotic systems.

8. References:

Harvey, I., Cliff, D., & Husbands, P. (1993). *Explorations in Evolutionary Robotics*. Retrieved from https://www.researchgate.net/publication/229091541_Explorations_in_Evolutionary_Robotics

Høvin, M., & Garder, L. M. (2006). *Robot gaits evolved by combining genetic algorithms and binary hill climbing*. Retrieved from https://www.researchgate.net/publication/220742643_Robot_gaits_evolved_by_combining_genetic_algorithms_and_binary_hill_climbing

Reeves, C. R. (2010). *Genetic Algorithms*. Retrieved from https://www.researchgate.net/publication/226462334_Genetic_Algorithms

Xu, Y. (2021). *Artificial intelligence: A powerful paradigm for scientific research*. Retrieved from https://www.sciencedirect.com/science/article/pii/S2666675821001041