Part One

Question 1

Fitness Plot of the Mice as a Function of the Genertion Count

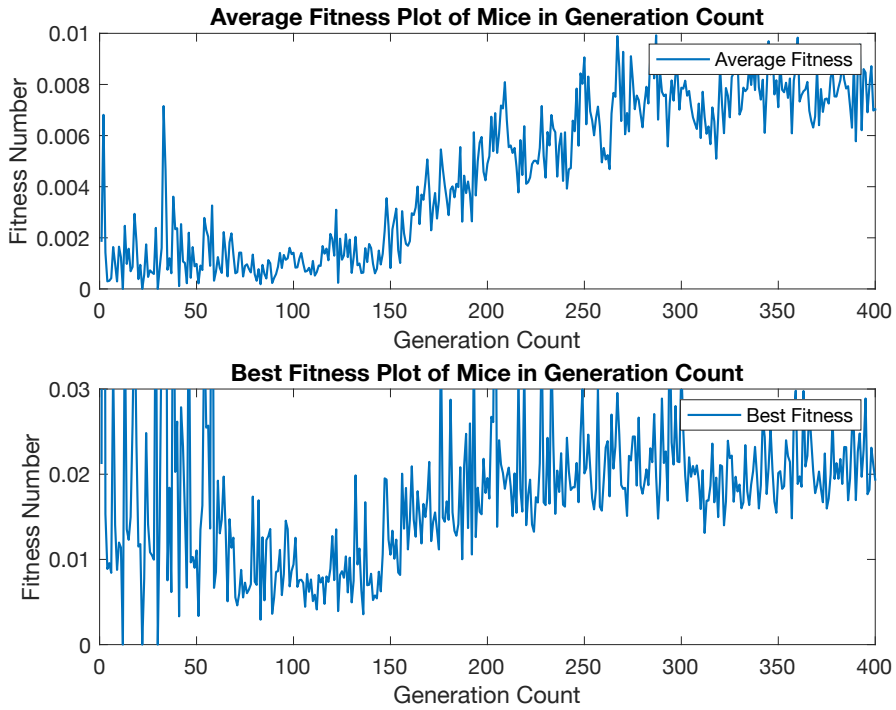

Figure 1. Average and Best Fitness versus Generation Count

The simulation results are exported in the `Question1/Mice.pop`, the data is extracted in `Question1/question_1.mat`, the plots are generated by MATLAB 2021b, the scripts for generating the plots is `Question1/Question1.m`.

Reasons Why Choosing This Range

The simulation was terminated after 3,000 generations to obtain enough data. Figure 2 shows the whole range of the data.

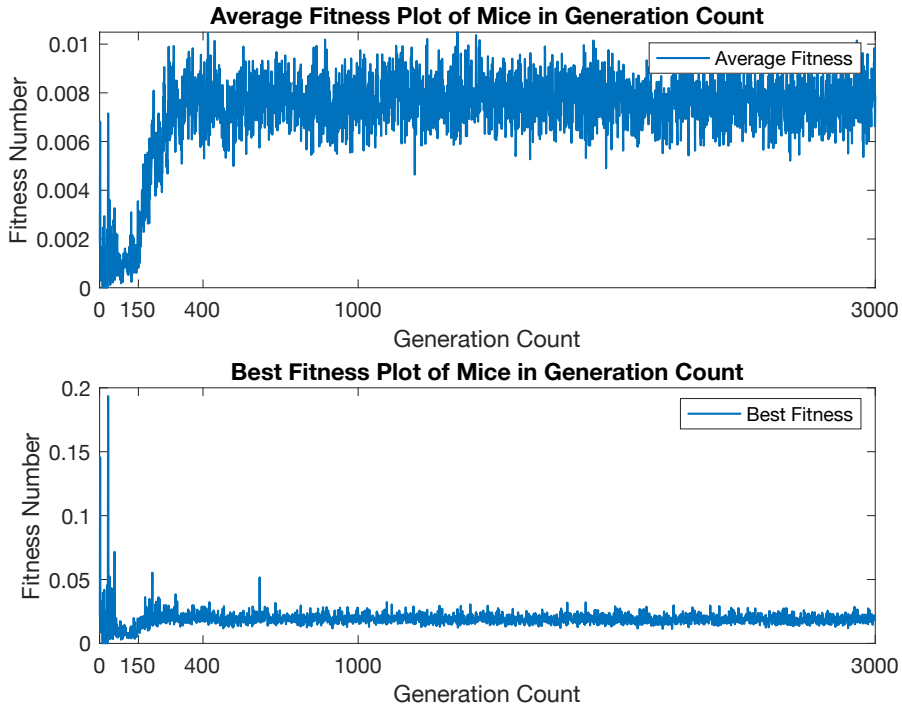

Figure 2. Average and Best Fitness Versus Generation Count Whole Range

- For the generation count range, In the generation domain from 0 to 150, the average fitness values are highly irregular. In the 150-400, the average fitness numbers increase non-linearly. After 400 generations, the average fitness of the mice becomes stable at around 0.008. The best fitness value also has similar characteristics. The generation range for both type of fitness are **0-400**
- For the fitness range. Considering the noise level and representive generation range.

Question 2

The mice evolves fatser, smarter and more efficient seeking the cheese over evolutionary time. The specific behaviours in explained in Table 1

Generation Count	Behaviour
0	The mice move randomly and slowly. Most of them are circling. Only a few mice are moving at a relatively faster speed in a relatively straight line. The mice rarely obtain the cheese.
300	The mice increase the speed. A few of the mice's movements show regular rules aiming at the cheese. However, the whole specie's behaviours are irregular. After finishing one cheese, the mice keep the previous direction. Some of the mice obtain the cheese.
500	The behaviour does not change a lot, but the moving speed of the mice increase a little. Some of the mice are competing for one cheese.
1000	All of the mice are moving at a relatively high speed. The mice are divided into many groups. Each group usually competes for one cheese. After finishing one cheese, the mice are susceptible to the nearest cheese and change its direction to obtain another cheese.
4000	The mice started to rotate on the spot to evaluate the distances of the cheese around it, which is a more efficient way to decide the behaviour.

Table 1. Mice Behaviours in Different Generations

Question 3

Discrition of the Fitness Function Used by the Mice

The fitness used by the Mice is in `Question1/mouse.cc` line 136-143.

```
        // The EvoMouse's fitness is the amount of cheese collected, divided by
        // the power usage, so a mouse is penalised for simply charging around
        // as fast as possible and randomly collecting cheese - it needs to find
        // its target carefully.
        virtual float GetFitness()const
        {
                return This.cheesesFound > 0 ? static_cast<float>(This.cheesesFound) / This.DistanceTravelled.as
        }
}
```

The fitness of each mouse has obtained this `GetFitness()` function. There are two variables in this function, `This.cheesesFound` indicates the cheeses found by the mouse in this generation simulation, and `This.DistanceTravelled.as` represents the distance the mouse travelled in this generation.

If the mouse found the cheese, its fitness value equals the amount of the cheese divided by the distance it travelled, or the fitness value equals 0.

To train the mice form an efficient way to find the cheese, the distance travelled by the mouse determines the power consumption influencing the fitness value.

Experiment with Different Fitness Functions Case 1

In case 1, the cheese number and the travled distance were changed irrelevant to the fitness function.

```
        virtual float GetFitness()const
        {
                return This.cheesesFound > 0 ? 0: 0;
        }
}
```

When the simulation started, the behaviour of the mice is similar to the original fitness function ones.

With the increment of the simulation generation count, the behaviour becomes more random and caotic. The performance of the mice deteriorated.

Experiment with Different Fitness Functions Case 2

In case 2, the fitness value and the amount of cheese collected have a first-order linear relationship. The fitness equals the amount of cheese divided by 100.

```
        virtual float GetFitness()const
        {
                return This.cheesesFound > 0 ? static_cast<float>(This.cheesesFound)/100 /: 0;
        }
}
```

When the simulation started, the behaviour of the mice is similar to the original fitness function ones.

With the increment of the simulation generation count, the behaviour becomes similar to the original fitness function, but the evololution speed decrease a lot. The performance of the mice deteriorated.

Experiment with Different Fitness Functions Case 3

In case 3, the fitness value and the travelled distance has a first-order linear relationship. The fitness equals the travelled distance of the mouse divided by 1000.

```
        virtual float GetFitness()const
        {
                return This.cheesesFound > 0 ? This.DistanceTravelled.as<float>()/10000 /: 0;
        }
}
```

When the simulation started, the behaviour of the mice was similar to the original fitness function ones.

With the increment of the simulation generation count, the speed of the mice increased a lot, and the mice moved directly rather than changing their direction. The performance of the mice deteriorated.

The fitness function determines the evolution's **trend** and **efficiency**.

Parameters in Genetic Algorithm

The fitness used by the Mice is in `Question1/mouse.cc` line 159-185.

```
class MouseSimulation : public Simulation
{
        GeneticAlgorithm<EvoMouse> theGA;
        Population<EvoMouse> theMice;
//      Group<Mouse> theMice;
        Group<Cheese> theCheeses;

public:
        MouseSimulation():
        theGA(0.7f, 0.05f),     // Crossover probability of 0.7, mutation probability of 0.05
//      theMice(30),            // 30 mice are in the population.
        theMice(30, theGA), // 30 mice are in the population.
        theCheeses(30)          // 30 cheeses are around at one time.
        {
                // We're using a rank selection method. Consult the BEAST
                // documentation for GeneticAlgorithm, the ar23 course slides or
                // a good book on GAs for more details.
                This.theGA.SetSelection(GA_RANK);
                // The ranking selection pressure is set to 2.
                This.theGA.SetParameter(GA_RANK_SPRESSURE, 2.0);

                This.SetTimeSteps(100);
```

```
                This.Add("Mice", This.theMice);
                This.Add("Cheeses", This.theCheeses);
        }
};
```

Parameters	Source Code	Influence
Crossover Probability	`0.7f`	The core in the evolution of nature is played by the crossover of biological genes. Similarly, The core in genetic algorithms is played by the crossover operator of genetic operations. A crossover is an operation in which parts of the structure of two parent individuals are replaced and recombined to create a new individual. By crossover, the search power of genetic algorithms is improved by leaps and bounds.
Mutation Probability	`0.05f`	The mutation operator is applied to change the value of a gene at some locus of a string of individuals in a population. When a genetic algorithm has approached the neighbourhood of the optimal solution through the crossover operator, this local stochastic search capability of the variation operator can be used to accelerate convergence to the optimal solution. The variation operator maintains population diversity to prevent immature convergence.
Selection Option	`GA_RANK`	The rank proportional selection of the invidual is used, preventing one or twooverwhelmingly fit individuals in a population from dominating the nextgeneration. So if in a population of 3 individuals you had scores of(2, 3, 400), rank selection would convert these to (0.167, 0.333, 0.5).
Ranking Selection Pressure	`GA_RANK_SPRESSURE`	When the pressure is low, we have relatively few randomly selected individuals, so that each individual faces less competitive pressure and thus has a higher probability of being selected.
Simulation Time	`SetTimeSteps(100)`	The simulation time determines the amount of the total behaviours in each generation

Table 2. Parameters in Genetic Algorithms

Question 4

The scripts for generating the plots and data is `Question4/Question4.m`, the data is extracted in `Question4/question_4.mat`.

Performance Criterions and Evaluation

The fitness function should be evaluated in different aspects, the evolution time and final performance.

In this section, the original fitness function is the original fitness function applied in the `Question4/mouse.cc` line 136-143. The fitness equals to the amount of cheese collected, divided by the distance it traveld.

```
        // The EvoMouse's fitness is the amount of cheese collected, divided by
        // the power usage, so a mouse is penalised for simply charging around
        // as fast as possible and randomly collecting cheese - it needs to find
        // its target carefully.
        virtual float GetFitness()const
        {
                return This.cheesesFound > 0 ? static_cast<float>(This.cheesesFound) / This.DistanceTravelled.as
        }
}
```

The new fitness function refers to `Question4/mouse_new.cc` line 136-14. The fitness equals to the amount of cheese collected.

```
        // The EvoMouse's fitness is the amount of cheese collected.
        virtual float GetFitness()const
        {
                return This.cheesesFound > 0 ? static_cast<float>(This.cheesesFound) : 0;
        }
}
```

Evolution Time

Evolution time is defined as the simulation's generation count to make the fitness value stable—the less generation amount it takes, the better the fitness function.

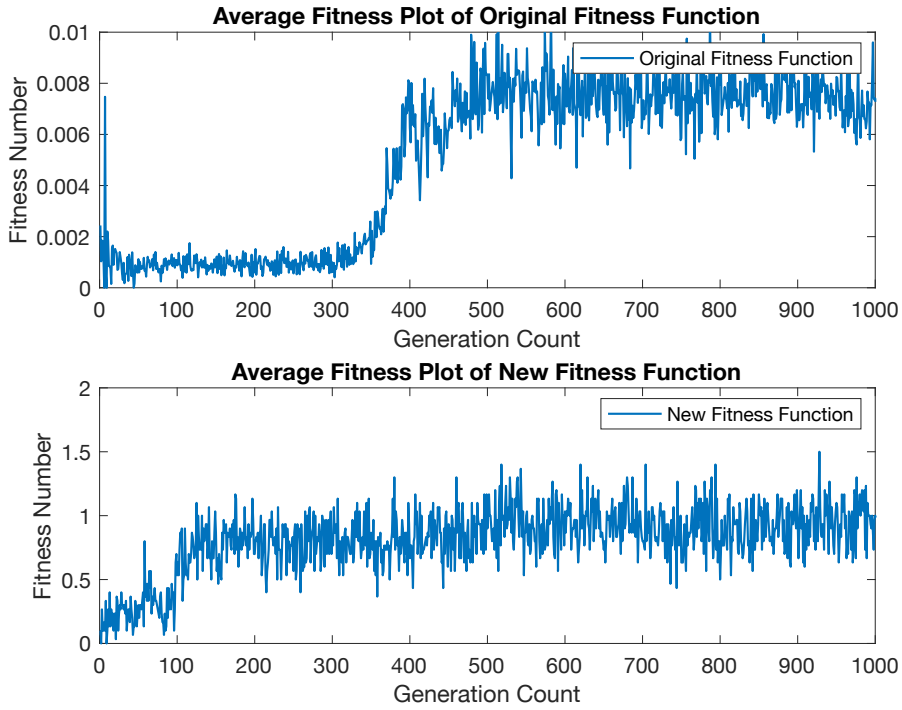

Figure 3. Convergenge of Different Fitness Function Comapre

It is easy to indicate the original fitness function can converge faster than the new fitness function, which means the original fitness function is betIt is easy to indicate that the original fitness function can converge faster than the new fitness function, which means it is better than the latest fitness function in this aspect.ter than the new fitness function in this aspect.

Stability of Average Fitness Value

This is defined by the average fitness value's stability and noise level. Healthier specie has more stable average fitness value.

Dispersion indicates the stability of the average fitness value. Because of the amplitude of average fitness values is not the same in different fitnesThis is defined by the average fitness value's stability and noise level. Healthier species has a more stable average fitness value.

Dispersion indicates the stability of the average fitness value. Because the amplitude of average fitness values is not the same in different fitness functions, the dispersion is measured by the SNR (Signal-to-noise ratio), assuming the mean value of the average fitness values is the original signal after convergence. The results are exported in the MATLAB command window.s function, the dispersion is measured by the SNR (Signal-to-noise ratio), assuming the mean value of the average fitness values is the original signal after convergence. The results are exported in the MATLAB command window.

```
SNR of original fitness function -0.1783
SNR of new fitness function -0.0682
```

The original fitness function has larger SNR, which means the appropriate noise level in the original fitness function's average fitness value is lower. The initial fitness function is better.

Cheese Allocation

This is defined by the standard deviation of all the fitness values after 1000 generations of evolution. Minor standard deviation indicates the fitness function has better performance.

However, I am not familiar with C++, I tried to modify the BEAST, but it was unsuccessful.

Different Sensor Configurations

In `Question4/mouse.cc`, line 111-120 the `EvoMouse` is applied in the simulation. The original `EvoMouse` is using the `NearestAngleSensor`.

```
        EvoMouse(): cheesesFound(0)
        {
```

```
                This.Add("angle", NearestAngleSensor<Cheese>());
// An alternative to the NearestAngleSensor is the Proximity Sensor, which
// gives less precise directional information, but does let the mouse know
// how far away the cheese is.
//              This.Add("proximity", ProximitySensor<Cheese>(PI/8, 80.0, 0.0));
                This.InitRandom = true;
                This.InitFFN(4);
        }
```

`NearestAngleSensor` is described in `Question4/sensor.h`. This sensor provides precise angular information but it does not let the mouse know how far the cheese is.

```
Sensor* NearestAngleSensor()
{
        Sensor* s = new Sensor(Vector2D(0.0, 0.0), 0.0);
        s->SetMatchingFunction(new MatchKindOf<T>);
        s->SetEvaluationFunction(new EvalNearestAngle(s, 1000.0));
        s->SetScalingFunction(new ScaleLinear(-PI, PI, -1.0, 1.0));

        return s;
}
```

I tested the `ProximitySensor` as well, the codes for different sensor configurations is below. The Angle and Range in Table 3 are the parameters with the same name in `ProximitySensor<Cheese>(Angle, Range, 0.0))`.

Serial Number	Angle	Range	File
1	2*pi	50	`Question4/mouse_sensor1.cc`
2	pi/3	50	`Question4/mouse_sensor2.cc`
3	pi/4	50	`Question4/mouse_sensor3.cc`

Serial Number	Angle	Range	File
4	pi/4	100	`Question4/mouse_sensor4.cc`
5	pi/4	200	`Question4/mouse_sensor5.cc`
6	pi/4	200	`Question4/mouse_sensor6.cc`
7	pi/6	50	`Question4/mouse_sensor7.cc`
8	pi/8	50	`Question4/mouse_sensor8.cc`

Table 3. ProximitySensor Configurations

```
        EvoMouse(): cheesesFound(0)
        {
//              This.Add("angle", NearestAngleSensor<Cheese>());
// An alternative to the NearestAngleSensor is the Proximity Sensor, which
// gives less precise directional information, but does let the mouse know
// how far away the cheese is.
                This.Add("proximity", ProximitySensor<Cheese>(Angle, Range, 0.0));
                This.InitRandom = true;
                This.InitFFN(4);
        }
```

`ProximitySensor` is described in `Question4/sensor.h`. This sensor provides less precise directional information but it let the mouse know how far the cheese is.

```
Sensor* ProximitySensor(double scope, double range, double orientation)
{
        Sensor* s = new BeamSensor(scope, range, Vector2D(0.0, 0.0), orientation);
        s->SetMatchingFunction(new MatchKindOf<T>);
        s->SetEvaluationFunction(new EvalNearest(s, Range));
        s->SetScalingFunction(new ScaleLinear(0.0, range, 1.0, 0.0));
```

```
        return s;
}
```

Sensor Experiment

All of the sensor experiments use the same original fitness function, so the mean value of the average fitness function is comparable. The sensor configuration with a higher mean value after convergence has better performance. SNR is also introduced to evaluate stability.

The results are exported in the MATLAB command window.

```
SNR of original sensor configuration -0.1783
SNR of sensor configuration 1 -0.4143
SNR of sensor configuration 2 -0.2334
SNR of sensor configuration 3 -0.3790
SNR of sensor configuration 4 -0.3124
SNR of sensor configuration 5 -0.4421
SNR of sensor configuration 6 -0.3589
SNR of sensor configuration 7 -0.2482314
SNR of sensor configuration 8 -0.4421
Mean Value of original sensor configuration 0.0077
Mean Value of sensor configuration 1 0.0009
Mean Value of sensor configuration 2 0.0031
Mean Value of sensor configuration 3 0.0033
Mean Value of sensor configuration 4 0.0024
Mean Value of sensor configuration 5 0.0021
Mean Value of sensor configuration 6 0.0027
Mean Value of sensor configuration 7 0.0022
Mean Value of sensor configuration 8 0.0021
```

Configuration Serial Number	Mean Value	SNR
Original	0.0077	-0.1783
1	0.0009	-0.4143
2	0.0031	-0.2334
3	0.0033	-0.3790
4	0.0024	-0.3124
5	0.0021	-0.44
6	0.0027	-0.36
7	0.0022	-0.25
8	0.0021	-0.44

Table 4. Different Configurations Sensor Experiment Result

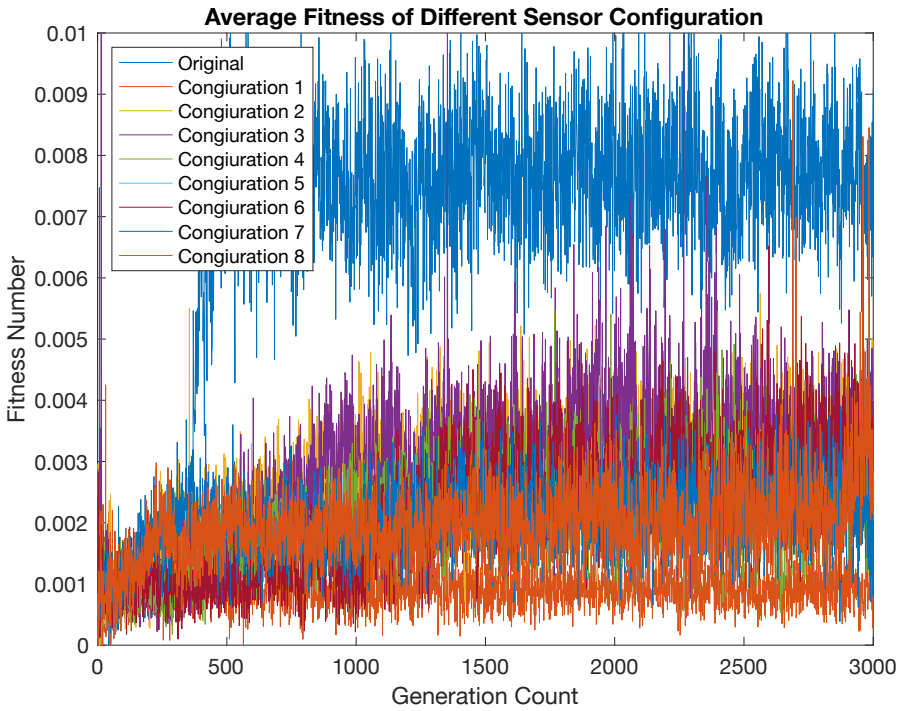

Figure 4. Average Fitness of Different Sensor Configuration

it is easy to indicate that, baesed on the mean values and amplitude plot, the final performancesthe `ProximitySensor` in these configurations are not as good as the `NearestAngleSensor`. Meanwhile, the `ProximitySensor` also has larger SNR, which means the `ProximitySensor`'s stability is better than others.

As for the `ProximitySensor`.

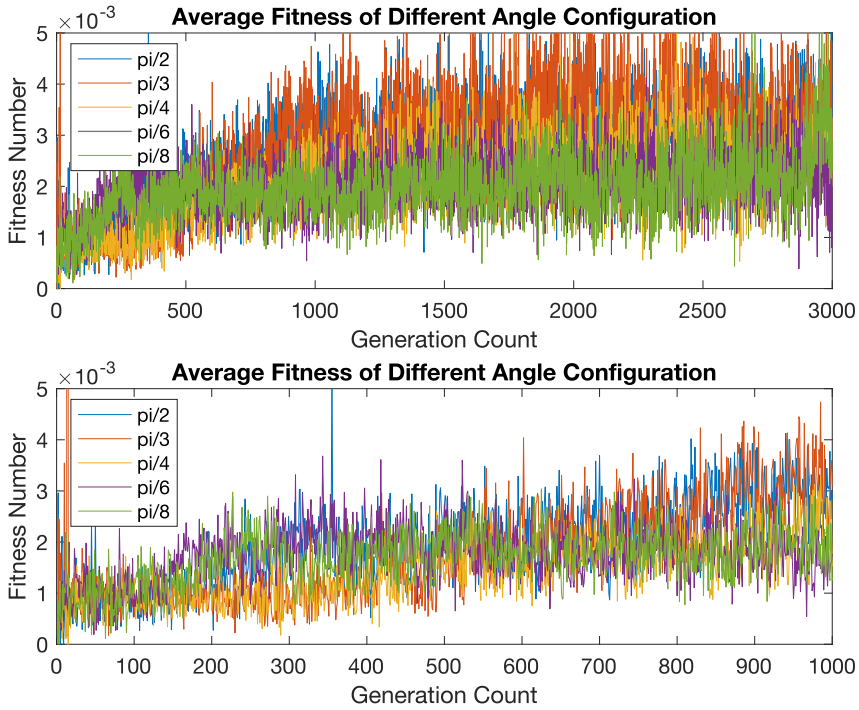

Figure 5. Average Fitness of Different Angle Configuration

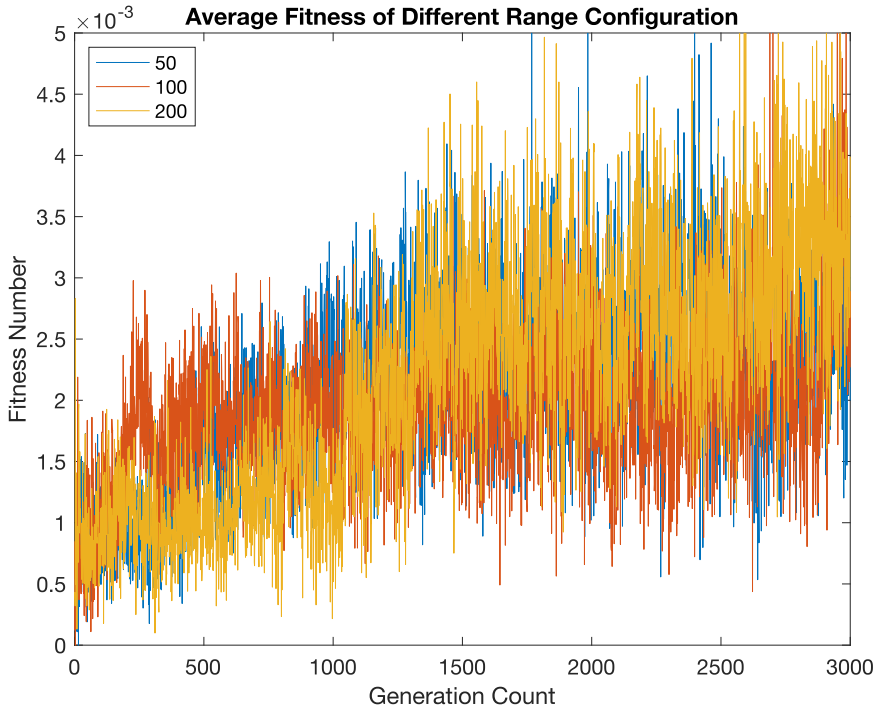

Figure 6. Mean Value of Average Fitness of Different Range Configuration

Based on Figure 5 and Table 4, it is easy to summarize that, with the increment of the angle, the performance of the sensor may increase, and converge faster. Based on Figure 6 and Table 4, larger range has better performance.

Reasons Behind Experiments

The main difference between these two sensor types is the mouse's field of view, which enables the mouse to sense and explore the cheese around it. The mouse can sense the cheese better and collect more cheese with a larger range or more extensive field of view.

Question 5

Review on Genetic Algorithms

Genetic algorithms emulate the natural evolutionary process to find the optimal solution, applying Darwinian evolutionary theory's principles of selection and variation. Directed selection, followed by undirected variation, reflects the fitness of each generation of individuals according to a fitness function that reflects the target, thus performing a selection operation followed by a genetic iteration that produces individuals with a new combination of genes.[1] [2]

The process of genetic algorithms is, in fact, a process similar to biological evolution in biology, in which at each generation in a genetic algorithm, individuals are selected according to the size of their fitness in the problem domain and, with the help of genetic operators for combinatorial crossover and subjective and objective variation, a population representing the new set of solutions is evolved. This process is performed cyclically until the optimisation criterion is satisfied. Finally, the last generation of individuals is decoded to generate a near-optimal key.[3]

Behaviour of the Mice

Collective behaviour results from the interaction and influence of each individual in the whole species. As for the mice simulation, the many small groups of mice usually compete for one cheese which is the best solution for the most vital individual or genetic clip. However, considering the reality, this behaviour is not the best approach to select the best genetic for the whole species, especially considering the equality of cheese allocation.

Mice Collective Behaviour

After the thousands of simulations, some of the mice start to follow other mice as groups to collect cheese rather than sense and explore the cheese independently. I regard this as evidence of collective behaviour.

Part Two

Question 6

Collegue Names: Yiqiao Wang, Rui Hou

The scripts for generating the plots and data is `Question6/question_6.m`, the data is extracted in `Question6/question_6.mat`.

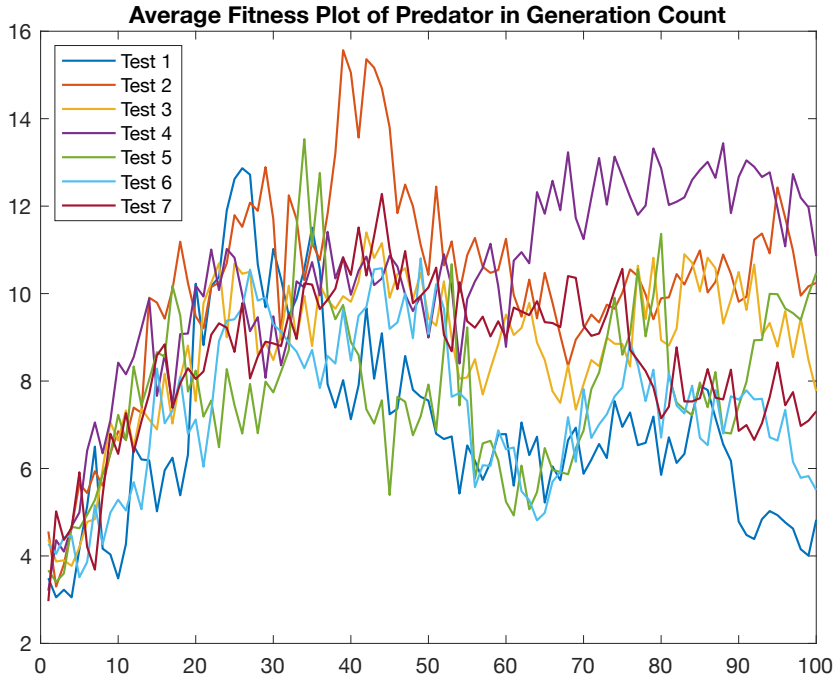

Figure 7. Average Fitness of Predator in Different Test

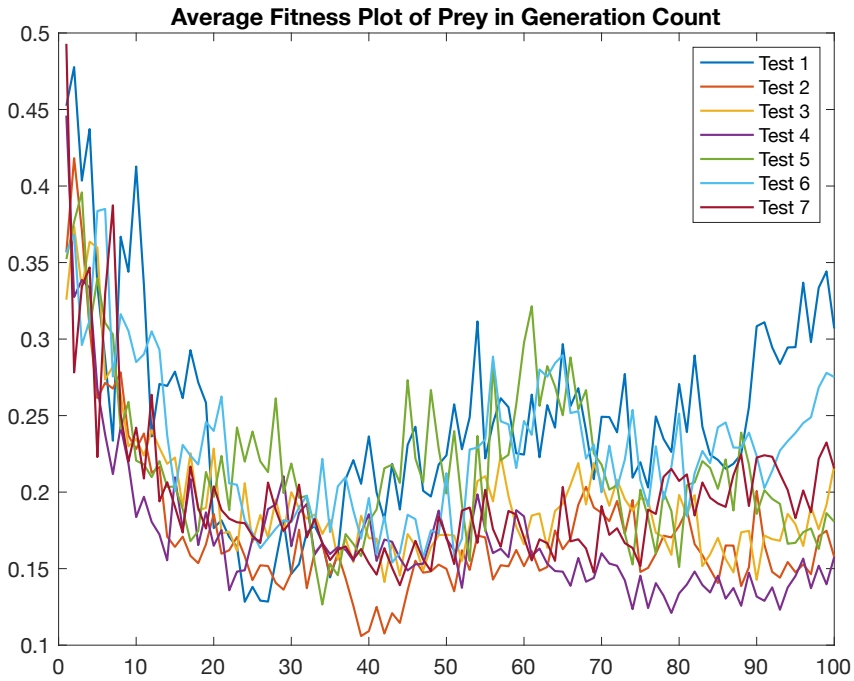

Figure 8. Average Fitness of Prey in Different Test

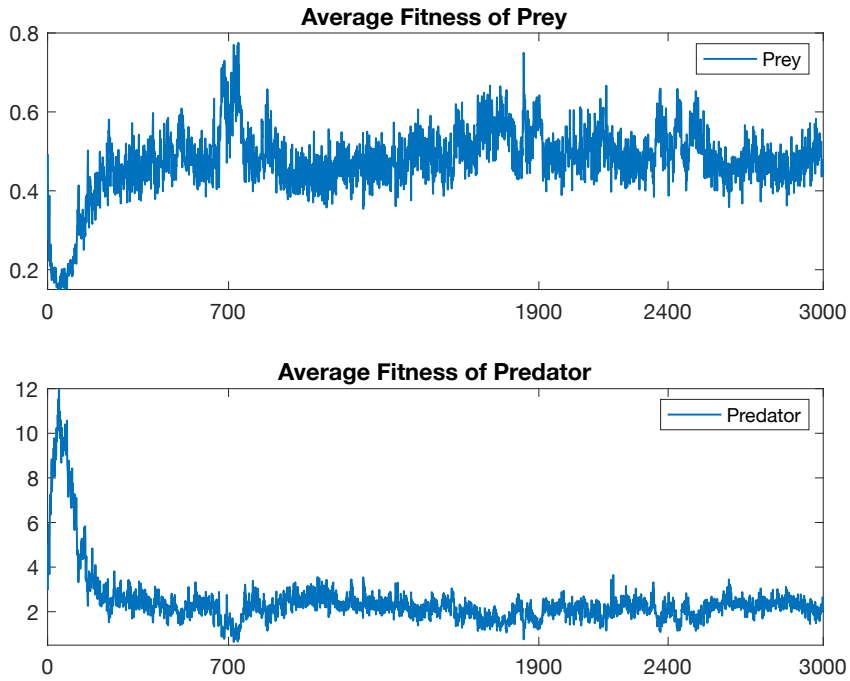

Figure 9. Average Fitness of Prey and Predator in Same Test

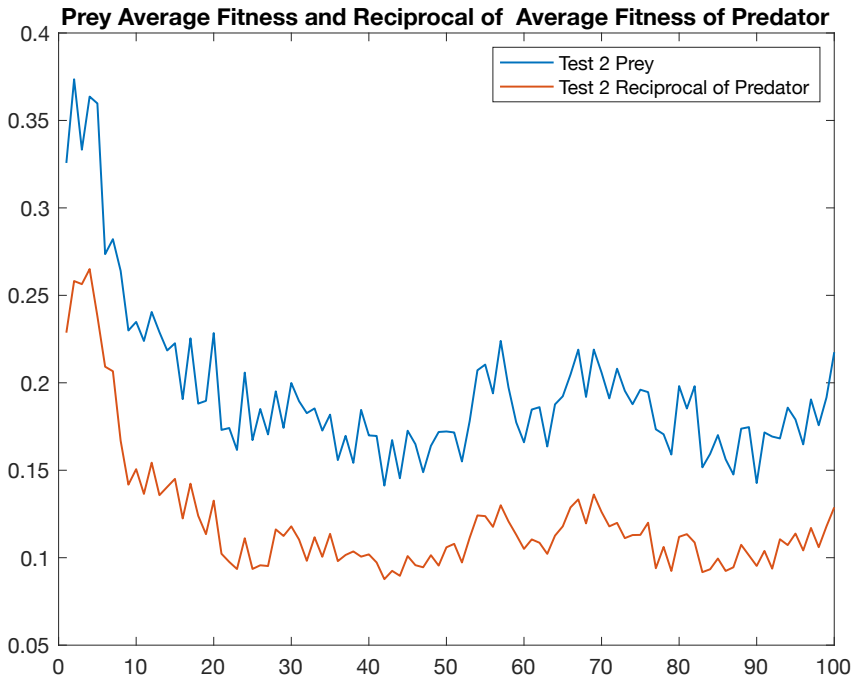

Figure 10. Prey Average Fitness and Reciprocal of Average Fitness of Predator

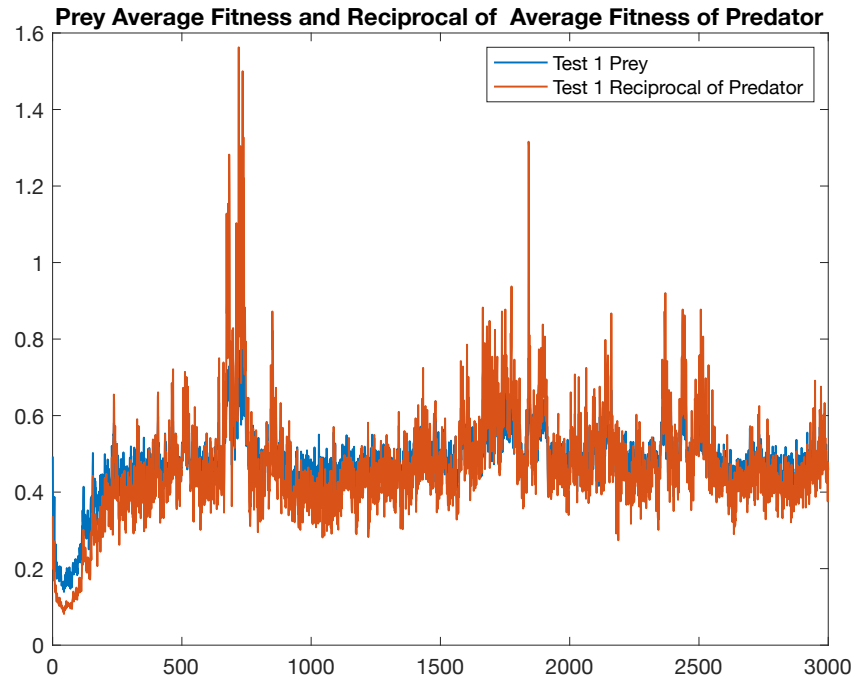

Figure 11. Prey Average Fitness and Reciprocal of Average Fitness of Predator

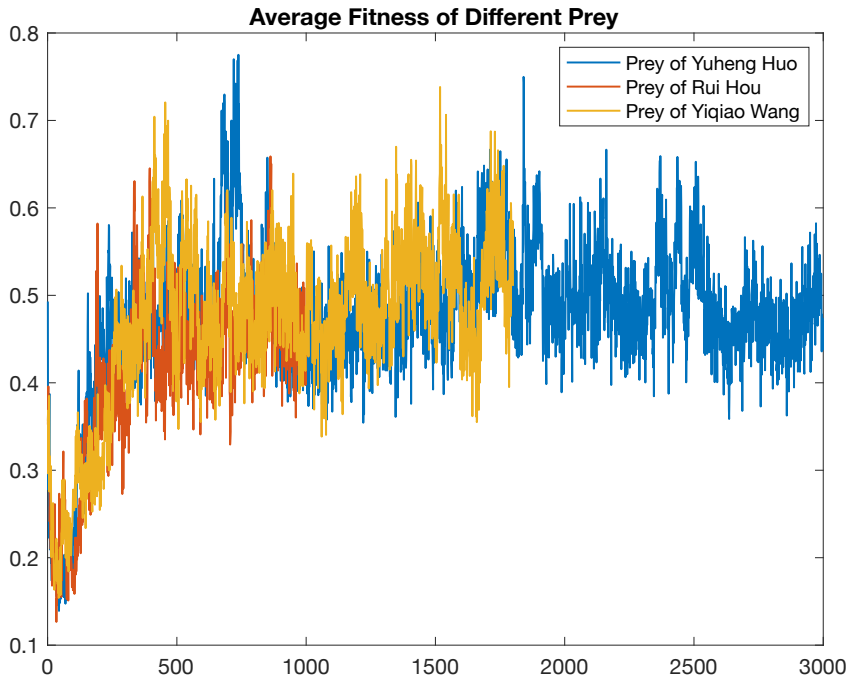

Figure 12. Different Student Results

Question 7

Generation Count	Predator Behaviour
0	The behaviour of the predator is highly irregular, even the prey is closely infront of it, the predator is still indifferent. Only some of the lucky predator may hitted by the prey.
10	A few predators start to change it direction to get closer to the preys, but the action to change the direction is very small, most of the predator continue to move straight.
50	The predator starts to circling around a large circle to scan more field to find the preys.

Generation Count	Predator Behaviour
100	The behaviour of predator does not change a lot.
1000	The behaviour mode of the predator does not change a lot, but the predator are more willing to twist around to scan more area.
5000	The predator starts to twisting around at a really high speed to evaluate the distance of different preys around it to decide which one to follow and catch, once the object is locked it will follow the prey closly.

Table 5. Predator Behaviours in Different Generations

Generation Count	Prey Behaviour
0	The behaviour of the prey is also highly irregular or stupid, some of the prey get caught by the predator, then it will revive and flash into another random place.
10	Most of the prey are moving around a circle, others are travling around. Some of the prey starts to avoid the prdator, the the action of changing the direction is also very small, but the action is very effective to stay alive.
50	Most of the prey are still moving around a circle, but the diameter of the circle get larger. But their behaviour to avoid the prdator is not as good as expected.
100	The prey starts to avoid the predator more efficiently.
1000	The prey can avoid the route of the predator efficiently. But there's also some of the preys get caught by the predator.
5000	The prey starts to twisting aroound to observe the predator around it, once the predator starts getting closer, it will speed up and run to get out of the field of view if the predator as soon as posibble.

Table 6. Prey Behaviours in Different Generations

Based on the behaviours above in this simulation rules, the prey and predator give pressure to each other. Based on the observed behaviour, both the predator and prey have become more intelligent. The prey's behaviours became more complementary to the predator. The behaviours of these species are in competitive arm races in many aspects, especially after thousands of simulations. The predator has become more aggressive and intelligent in detection, and the prey has become more sensitive and intelligent in anti-detection and escaping.

So they are in co-evolution.

Question 8

The behaviours of the agents are determined by the fitness functions in `Question6/chase.cc`.

As for the Prey, the fitness function is in lines 47-50. The code indicates that fitness equals the reciprocal of getting caught by the predator. Healthier prey gets caught less.

```
        float GetFitness()const
        {
                return 1.0f / static_cast<float>(timesEaten);
        }
```

As for the Predator, the fitness function is in line 95. Its fitness equals to the amount of the prey it catch. Healthier predator catch more preys.

```
        float GetFitness()const { return preyEaten; }
```

Based on the fitness functions above, the average values of the fitness should have similar characteristics to the reciprocal relationship.

In Figure 9, Figure 10 and Figure 11 above. It is easy to indicates that the averge fitness between the prey and predator have reciprocal relationship.

In Figure 9, when the average fitness of prey increases, the average fitness of the predator will decline, and vice versa, especially in the generation count equals around 700, 1900 and 2400. The soundness of these two species is very sensitive to each other. This means the prey and the predator are in competitive arm races and co-evolution.

Part Three

Question 9

In perceiving the purposive target object, the subject produces a cognition that the other party can process, process, or has the corresponding function, and is referred to as intelligence[4]. From an informatics perspective, this purposeful processing of intelligence is also a process of processing information, which derives its processing logic from the extraction and application of objective laws and drives.[5] Intelligence can therefore also be seen as a manifestation of the use of knowledge.[6] Intelligence can also be used as a general term for developing things according to a system of inertia without external forces. In contrast, purposeful processing changes the original development direction as if some power is generated. Different items have different processing logic. They have different purposes, ranging from the overly simple, such as conditioned reflexes, to the overly complex, where simple logics are stitched together, linked, nested, and combined to create a variety of abilities, such as perception, memory, imagination, thinking, attention, rallying, cohesion, execution, etc. However, the generalised titles can be summarised[7].

There are categories of intelligence sizes, and the various intelligence can co-exist, depend on each other, complement each other, be connected, or even be mutually exclusive.[8] Any combination of them constitutes a wide range of intelligence. From automatic doors with a single function to automated production lines with complex processes; from micro-organisms with the capacity to grow, reproduce and adapt, to human beings of all kinds, including groups of people, such as units, groups, organisations, nations, societies and human races, and even elementary particles and cosmic stars that can bounce off, deform or disintegrate on impact[6]. From a rigid body with intelligence infinitely close to zero to the quantification of the extremes of each human ability, a full-dimensional radar map can be constructed, then the hypothetical human, which sets all the extremes of human power, represents the highest standard of the current level of human intelligence, which is also the highest standard of intelligence of known species, and can be tentatively called standard intelligence[9]. Traditional intelligence will continue to improve as humans explore, dig, and accumulate new knowledge.

Artificial intelligence has been given the ultimate goal of creating a level of intelligence like that of a human being[10]. This is where scientists from different fields converged. Starting with the structure, function, and role of the human brain, they formed three mainstream schools of thought, corresponding to connectionism, symbolism, and behaviourism.

Brain-like intelligence is a continuation of connectionism, a vision of artificial intelligence proposed in the late 1980s. They hoped to study the working mechanism of the human brain and simulate a robot with the same ability to think and learn as humans[11]. It currently relies on two main techniques: deep learning and reinforcement learning.

Cognitive intelligence was developed out of computing and is pretty much a continuation of symbolism. They believe that there are three stages to achieve artificial intelligence: computational intelligence, perceptual intelligence and cognitive intelligence, where cognitive intelligence mainly addresses the ability to understand and actively think, and is currently popularly divided into language comprehension, analytical reasoning and personality emotion.[12]

General intelligence is machine intelligence with general human intelligence and can perform any intellectual task that humans can perform. It aims to pass the Turing test, a continuation of behaviourism and is currently mainly in the academic discussion stage and simulation.[13]

The strength of intelligence is relative. As long as there is self-learning ability, it can change from weak to strong, so vital intelligence does not mean strong intelligence but refers explicitly to acquiring knowledge actively. **The prey and the predator has shown their ability to learn and develop intelligence**, although their intelligence is not as vital as the species in reality. Our world is a collection of countless factors and species. However, **this level of intelligence is limited**, all of the results can be simulated and predicted because of the limiting factor in the simulation creature in the simulation world. The actual intelligence is strongly connected with free will and actual random with no limitation. Before discovering quantum mechanics, some of the scientists believed all of our behaviour could be calculated and simulated with mathematical modelling and approach, so the whole world is running with a fated programme and pattern. But for free will in the metaphysical sense, that contradicts determinism. Assuming a determinism like Laplace's demon, where future events (including all human actions) are uniquely determined by past events, free will in the metaphysical sense is impossible. Therefore, such free will is possible if determinism can be shown to be false. And quantum mechanics can then be used to introduce non-determinism. The indeterminacy of the quantum is inherent to it, which is fundamentally different from the indeterminacy of the dice.

References

[1] Debreuve, E., et al. "Using the shape gradient for active contour segmentation: from the continuous to the discrete formulation." *Journal of Mathematical Imaging and Vision* 28.1 (2007): 47-66.

[2] Sridevi, T., and S. Sameen Fatima. "Digital image watermarking using genetic algorithm in DWT and SVD transform." (2013): 485-490.

[3] Mitchell, Melanie. *An introduction to genetic algorithms*. MIT press, 1998.

[4] McQuillan, Jeff, and Lucy Tse. "Child language brokering in linguistic minority communities: Effects on cultural interaction, cognition, and literacy." *Language and Education 9.3* (1995): 195-215.

[5] Lumia, Ronald, John Fiala, and Albert Wavering. "I ntelligent C ontrols Group National Bureau of Standards.

[6] Bradford, Gwen. "Knowledge, achievement, and manifestation." Erkenntnis 80.1 (2015): 97-116.

[7] Castree, Noel. "Neoliberalising nature: the logics of deregulation and reregulation." *Environment and planning* A 40.1 (2008): 131-152.

[8] Neisser, Ulric. "The concept of intelligence." *Intelligence* 3.3 (1979): 217-227.

[9] MLA Deary, Ian J., and Peter G. Caryl. "Neuroscience and human intelligence differences." *Trends in Neurosciences* 20.8 (1997): 365-371.

[10] PK, FATHIMA ANJILA. "*What is Artificial Intelligence?." "Success is no accident. It is hard work, perseverance, learning, studying, sacrifice and most of all, love of what you are doing or learning to do".* (1984): 65.

[11] Velik, Rosemarie. "AI reloaded: objectives, potentials, and challenges of the novel field of brain-like artificial intelligence." BRAIN. *Broad Research in Artificial Intelligence and Neuroscience* 3.3 (2012): 25-54.

[12] MLA Côté, Stéphane, and Christopher TH Miners. "*Emotional intelligence, cognitive intelligence, and job performance.*" *Administrative science quarterly* 51.1 (2006): 1-28.

[13] Spearman, Charles. "" *General Intelligence*" Objectively Determined and Measured." (1961).