Evaluation and Conclusion

Contents

[Testing 2](#_Toc165324952)

[Algorithm Effectiveness Testing 2](#_Toc165324953)

[Evaluation Process 10](#_Toc165324954)

[Discussion 10](#_Toc165324955)

[Goal Review 10](#_Toc165324956)

[Reflection 13](#_Toc165324957)

[Learning Algorithm Implementation Process 13](#_Toc165324958)

[Milestone Planning Approach 14](#_Toc165324959)

[Conclusion 15](#_Toc165324960)

[References 16](#_Toc165324961)

# Testing

### Algorithm Effectiveness Testing

Aim

The aim of these tests is to evaluate the effectiveness of the implemented algorithms against neural networks that have no learning capabilities.

**Default Environment setup**:

|  |  |
| --- | --- |
| Variable | Value |
| Grid Size | 1,000 x 1,000 |
| Seed | 1 |
| Solid Density | 5% |
| Nutrient Density | 100% |
| Lava Density | 0% |
| Poison Density | 0% |
| Number of Updates | 50000 |
| Should Agents Learn | False |
| Nutrient Values | |
| Ticks Between Spawns | 30 |
| Number of Attempted Spawns | 2,000 |
| Max Spawned Value | 5.0 |
| Min Spawned Value | 0.5 |
| Agent Values | |
| Number of Hidden Layers | 0 |
| Ticks Between Spawns | 40 |
| Number of Attempted Spawns | 5 |
| Maximum Number of Agents for New Spawns | 100 |
| Minimum Species Diversity | 50 |
| Mutation Chance | 5% |
| Energy Required to Reproduce | 200.0 |
| Starting Energy | 100.0 |
| Reward Values | |
| Randomize agent values | False |
| Nutrient Sensitivity | 0.1 |
| Health Sensitivity | 0.1 |
| Energy Sensitivity | 1.0 |
| Lava Sensitivity | 0.1 |
| Poison Sensitivity | 0.1 |
| Punishment Sensitivity | 10.0 |

Effectiveness Testing Method

1. Apply variable changes in dashboard.
2. Set ‘Should Simulation Stop’ to true.
3. Set ‘Updates to simulate’ to 50,000.
4. Ensure variables not changed by test are the same as variables found in Default Environment Setup.
5. Click “Begin Simulation”
6. Once simulation completes, a tone will sound, and simulation will pause.
7. Record the following data:
   1. **Agents Above 1.0 Efficiency Graph**
   2. **Species Count Graph**
   3. **Average Efficiency Graph**
   4. *Number Of Updates for First Population Peak (if applicable)*
   5. *Number Of Successful Agents at Peak (if applicable)*
   6. *Number Of Updates for Average Efficiency to Pass 1.0 (if applicable)*
   7. *Average Nutrient Sensitivity Graph (if applicable)*
   8. *Average Health Sensitivity Graph (if applicable)*
   9. *Average Energy Sensitivity Graph (if applicable)*
   10. *Average Lava Sensitivity Graph (if applicable)*
   11. *Average Poison Sensitivity Graph (if applicable)*
   12. *Average Punishment Graph (if applicable)*
8. Click “Reset Simulation” and uncheck “Pause Simulation” to prepare for next test.

Test 1

**No algorithm, no hazards (control)**

|  |  |
| --- | --- |
| Adjusted Values | |
| Default | **No Change** |

**Results:**

A graph on a black background

Description automatically generated

A screen shot of a graph

Description automatically generated

A screen shot of a graph

Description automatically generated

|  |  |
| --- | --- |
| Data Collected | Data Value |
| Number Of Updates for First Population Peak | 33,035 Updates |
| Number Of Successful Agents at Peak | 756 Agents |
| Number Of Updates for Average Efficiency to Pass 1.0 | 11,820 Updates |
| Notes | |
| While the average efficiency did eventually pass 1.0, the population growth was quite slow and only peaked near the last third of the simulation. | |

Test 2

**Simple algorithm, no hazards, no random sensitivity values**

|  |  |
| --- | --- |
| Adjusted Values | |
| Should Agents Learn | **True** |

Results:

A graph on a black background

Description automatically generated

A screen shot of a graph

Description automatically generated

A screen shot of a graph

Description automatically generated

|  |  |
| --- | --- |
| Data Collected | Data Value |
| Number Of Updates for First Population Peak | 20,871 Updates |
| Number Of Successful Agents at Peak | 2,498 Agents |
| Number Of Updates for Average Efficiency to Pass 1.0 | 5,843 Updates |
| Notes | |
| Compared to no learning, the agents reached a successful population peak 3.3x higher than agents without learning implemented, with the peak occurring 1.6x faster. This peak takes less time to collapse, however, most likely due to the amount of nutrients available being depleted quicker. | |

Test 3

**Simple algorithm, no hazards, random sensitivity values**

|  |  |
| --- | --- |
| Adjusted Values | |
| Should Agents Learn | **True** |
| Randomize Agent Values | **True** |

Results:

A graph with a line going up

Description automatically generated

A graph with a line

Description automatically generated

A screen shot of a graph

Description automatically generated

A screen shot of a computer

Description automatically generated

A graph on a screen

Description automatically generated

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

|  |  |
| --- | --- |
| Data Collected | Data Value |
| Number Of Updates for First Population Peak | 26,975 |
| Number Of Successful Agents at Peak | 2,186 |
| Number Of Updates for Average Efficiency to Pass 1.0 | 12,088 Updates |
| Notes | |
| Since only one seed is used, we cannot make any assumptions on what the best value for each sensitivity is. However, what this does highlight is that every time a population peak crashes and a new peak begins to rise, it effectively signifies a new “age” for the agents, with a new agent strategy coming out on top. We can see this with the sudden change in average sensitivity values at around 38,000 updates coinciding with the end of the first population peak and the start of the second population peak. | |

Test 4

**No Algorithm, Minor Hazards**

|  |  |
| --- | --- |
| Adjusted Values | |
| Should Agents Learn | **False** |
| Lava Density | **0.01** |
| Poison Density | **0.05** |

Results:

A graph on a screen

Description automatically generated

A graph on a screen

Description automatically generated

A screen shot of a graph

Description automatically generated

|  |  |
| --- | --- |
| Data Collected | Data Value |
| Number Of Updates for First Population Peak | 38,959 Updates |
| Number Of Successful Agents at Peak | 791 Agents |
| Number Of Updates for Average Efficiency to Pass 1.0 | 12,489 Updates |
| Notes | |
| The addition of hazards that agents need to avoid drastically reduces the speed at which successful agents are found. | |

Test 5

**Simple Algorithm, Minor Hazards**

|  |  |
| --- | --- |
| Adjusted Values | |
| Should Agents Learn | **True** |
| Lava Density | **0.01** |
| Poison Density | **0.05** |

**A graph on a black background

Description automatically generated**

A graph with blue lines

Description automatically generated

A screen shot of a graph

Description automatically generated

|  |  |
| --- | --- |
| Data Collected | Data Value |
| Number Of Updates for First Population Peak | 31,045 Updates |
| Number Of Successful Agents at Peak | 1,967 Agents |
| Number Of Updates for Average Efficiency to Pass 1.0 | 14,004 Updates |
| Notes | |
| Learning algorithm increases the speed at which the first peak is reached 1.25x faster when hazards are present, with a max peak value 2.5x higher. Even with hazards present, the implementation of a learning algorithm causes a peak to be reached faster than the control that has no hazards present. | |

# Evaluation Process

The original objective of this project was to create a proof of concept for a procedurally generated, continuously evolving ecosystem built around neural networks. For my evaluation, I will be listing the goals set at the beginning of development, if those goals were met, what went well during the implementation of those goals and what could be improved in the future.

For reflection and discussion on the process of implementing this proposal I will be using the Gibbs Reflective Cycle (The University of Edinburgh, 2020).

# Discussion

## Goal Review

|  |  |
| --- | --- |
| Goal | A 2D grid with cells that contain data determining how much food is present, how high the terrain is and what scents are present. |
| Was Goal Met? | Yes |
| What Went Well | A simple grid of variable size was successfully created, with each cell containing the required values for nutrients, terrain, and agents. The grid can be edited and modified by agents if required, and adding and removing cell data types is relatively simple. |
| What Could be Improved | Only one type of Cell class is used, meaning that more complex cells will require the creation and implementation of child classes. Cell colour selection is also quite fragile because of this, with the cell’s colour being determined by the order of a series of if-else statements.  Scents have not been implemented, which would add more complexity and unique situations for agents to respond to, allowing for more effective training. |

|  |  |
| --- | --- |
| Goal | An Agent class that contains multiple components that determine what the agent can sense and what actions the agent can take. |
| Was Goal Met? | Yes |
| What Went Well | Agents are very flexible, with the ability to add collections of sensors and actions easily. Controlling the agent is simple and intuitive, with three functions for movement and intuitive functions for modifying values such as health and energy. |
| What Could be Improved | Currently agent colours are completely random. An improvement for visibility would be to base an agent’s colour on the collection of inputs and sensor that makes up the agent’s brain. This would also allow agents to change colour over time after a mutation.  More mutations could be implemented, as current mutations do not impact an agent’s chance of survival enough to be noticeable in the long run. |

|  |  |
| --- | --- |
| Goal | A Brain that will be attached to each agent, which is a neural network that procedurally learns based on different rewards determined by the agent. |
| Was Goal Met? | Yes |
| What Went Well | The brain implementation is modular, with networks being very easy to generate and process. The brain also does not rely on any third-party libraries outside cista for of serialization, so transferring the code and implementing it into other projects should be easy to do. |
| What Could be Improved | Implemented learning algorithms only work on networks that contain no hidden layers. Investigating the cause for this and improving the algorithms to work with hidden layers, as well as improving hidden layer flexibility, would allow for agents to learn in more complex environments and improve future behaviours.  A lot of time during development was dedicated to investigating the implementation of these networks, during which time I relied entirely on online sources and academic papers. This time could have been more streamlined and improved if I had approached academics in the field directly for input and potential paths of research that would help me achieve my goal more effectively. |

|  |  |
| --- | --- |
| Goal | Agents will be able to reproduce and create children, with these children having a chance to evolve and be slightly different to the parent agent. |
| Was Goal Met? | Yes |
| What Went Well | Implementation was straightforward, with new agents being copies of the parent agent with potential mutations. |
| What Could be Improved | The current selection of mutations is limited and only include adjustments to the brain. If variables such as agent colour and scents were implemented, further mutations could include adjustments to these in order to increase system diversity.  Currently all mutations have the same chance of happening. An improved system would allow for certain mutations to be more common than others, for example colour mutations would be more common than network mutations. |

|  |  |
| --- | --- |
| Goal | Procedurally generated 2D terrain that is generated at the start of the simulation. This terrain can affect how the agents move and what kind of food can grow in certain regions. |
| Was Goal Met? | Mostly |
| What Went Well | A world state grid was created with both terrain and hazards implemented in an effective way. Agents can easily interact with the cell that they currently occupy. |
| What Could be Improved | Food growth is currently not affected by other terrain variables, meaning that the overall diversity of the grid does not change too much from region to region. Terrain height generation is also very basic, so features such as cliffs or mountains do not exist, with the height map currently being simple rolling hills. Both aspects could be improved upon to improve the diversity of the world state as a whole and encourage more diversity within the agent population.  I did not prioritise these aspects of implementation enough, resulting in a world state where a single agent species could survive in any region of the world and making the testing of a diverse ecosystem less effective. |

|  |  |
| --- | --- |
| Goal | A UI that can display selected Agents and their current genetic makeup, including current brain weights and components. |
| Was Goal Met? | Yes |
| What Went Well | Utilizing ImGui, the implemented UI is extremely flexible and allows for very easy variable control. Utilising ImGui’s DrawList functionality, I was also able to implement a very intuitive and easy to read neural network graph that could be used in other neural network applications as well. |
| What Could be Improved | For a more polished application, it would be better if the dashboard and the world state view were located on the same window to remove the need of forcing the world state window to always be on top of all other windows. |

|  |  |
| --- | --- |
| Goal | Highly dynamic training rules that change how different agents learn and adapt to their environment, without requiring manual tuning to find an effective ruleset. |
| Was Goal Met? | Yes |
| What Went Well | As shown in testing, the implemented algorithm vastly improves the agent’s effectiveness in the world, and with the ability to randomly select the reward value sensitivities, it would be possible to run multiple simulations and have different strategies evolve. |
| What Could be Improved | The current implementation only words with simple networks that contain an input layer and an output layer. The addition of hidden layers causes the algorithm to break down and agent learning grinds to a halt, in some regards becoming worse than purely random networks. |

## Reflection

### Learning Algorithm Implementation Process

Description

The implementation of the learning algorithm was a continuous process across the second half of the development process. The learning algorithm was implemented in multiple ways before the current version of the algorithm was finalized, with two versions of the final algorithm being present in the final program but only one being actively used for demonstration purposes.

Feelings

Overall, I think I vastly underestimated how long the algorithm would take to develop and how difficult it would be to finalize. While I am happy that the final version of the algorithm does function better than agents with no algorithm implemented, the development process could have been much smoother. During development, there were concerns that the final resulting algorithm would not match the requirements or expectations set in the proposal, which ended up causing more stress than expected.

Evaluation

The tests implemented to evaluate algorithms during development were easy to run and only took 10-20 minutes for a useful output to aid in improvements, however debugging and detecting potential errors within the code was quite challenging due to the extremely large sample size that would need to be checked to identify issues that the algorithm may have caused. Thankfully, reproducing issues once they were spotted was relatively easy thanks to the seeded random number generator, but debugging a single agent in a sea of 1,000 agents proved tricky.

Analysis

While my implemented visualization tools helped a lot with evaluating the status of the neural networks, there was no framework set up to directly test the code as development progressed. This meant that as development progressed and changes were made to the code, certain systems may change behaviour in unexpected ways that may cause subtle bugs that go undetected, which is something that Unit Testing (Amazon, 2024) aims to solve.

Conclusion

During this process I learned that unit testing could be a very useful tool for streamlining and speeding up the development of complex systems with many intermingling subsystems. On top of this, requesting assistance from my piers with experience on similar systems to avoid common pitfalls and challenges that could easily be solved with an additional perspective is very useful. Finally, implementing useful visualization tools early on in development helps greatly with debugging and ensuring the system is working as intended.

In future projects I will ensure to consider unit testing from the very beginning of development, ensuring that systems are built around tests that can catch issues early. I will also communicate with my piers and make a list of people with experience in the field that have agreed to answer questions if they arise.

### Milestone Planning Approach

Description

While developing the program I opted to use a Milestone planning method (Andersen, 2006) in which goals were set using MoSCoW prioritization (International Institute of Business Analysis, 2009) in order to determine what state the project should be in for the given milestone.

Feelings

By using milestones to track my progress, I felt that I was able to pace myself and keep on top of what was required for the project at that time. By breaking the project up into sections, I was able to avoid a lot of stress during the late stages of the project as a good foundation had been created by meeting the milestones. I found that having multiple smaller deadlines is much less intimidating than one single giant deadline at the end of development.

Evaluation

Milestones allowed for more focused development on each system within the program and greatly reduced the potential stress of large final deadlines. However, as development progressed, certain systems that were planned for one milestone ended up requiring more time to implement than expected, which caused a knock-on effect with later milestones becoming more bloated with vital systems that were carried over from the previous milestone. Due to MoSCoW prioritization, it was easier to handle these carried over tasks by removing lower priority tasks from the milestone.

Analysis

Since this was a solo project, milestones and MoSCoW worked well due to the low chance of project requirements changing mid-development as may happen with larger projects ordered by clients. This meant that tasks and goals could be pre-determined at the start of development. However, if this was a larger project, pre-determining tasks like this may lead to issues with adapting and adjusting to requirement changes down the line. A better approach in larger scale projects with a less defined end goal would be an Agile approach (Laoyan, 2024), and milestone tasks and requirements can be determined as the new milestone begins, rather than at the very start of development.

Conclusion

During this process I learnt that pacing yourself during development is an important skill to have to prevent the project from becoming overwhelming. By defining clear and concise goals and milestones, it also prevents “feature creep” in which new features are slowly added over time without any clear goal in mind, causing the project to become bloated and unmanageable. I also learnt more about setting and upholding deadlines and prioritizing what systems need to be done to produce a high-quality program.

In future projects, I will aim to integrate a more agile methodology to accommodate potential changes in requirements, as well as wait until the previous milestone is almost complete before planning and organising the tasks required for the next milestone.

# Conclusion

Overall, the project was successful. While there are plenty of areas that could be polished and improved, the world state provided a challenging enough situation for survival, allowing for multiple learning strategies to be viable for survival while showing that relying purely on survival or the fittest and no learning was less than ideal. The testing environment developed is well suited for future development and research into different learning algorithms and network architectures, with the data display and input variables being easily maintained and added to thanks to the ImGui UI library (Cornut, 2024).

Further development could include the adaptation of the neural network and brain systems into a publicly available library that could potentially be used by other developers for research and implementation into their own projects, and further research can certainly be done into developing and testing a more complex and effective learning algorithm that could potentially handle networks with multiple hidden layers.

The networks used could also be improved to utilise machine vision, allowing the agents to “see” the world around them and make a system where camouflage can be a viable and useful survival tool instead of just a decoration.

Finally, there is clear evidence in testing that the implemented learning algorithm can boost and improve the development and propagation of an artificial ecosystem, with this ecosystem moving through phases and ages where different strategies become dominant at different times. This is the kind of behaviour that could improve the replayability of exploration games, with planets adapting and evolving over time allowing for documentation of both current system and system history.

# References

Amazon. (2024, April 27). *What is Unit Testing?* Retrieved from AWS: https://aws.amazon.com/what-is/unit-testing/#:~:text=Unit%20testing%20is%20the%20process,test%20for%20each%20code%20unit.

Andersen, E. S. (2006). *Milestone planning--a different planning approach.* Retrieved from Project Management Institute: https://www.pmi.org/learning/library/milestone-different-planning-approach-7635#:~:text=Milestone%20planning%20is%20focused%20on,developed%20specifically%20for%20milestone%20planning.

Cornut, O. (2024, April 26). *Dear ImGui*. Retrieved from Github: https://github.com/ocornut/imgui

International Institute of Business Analysis. (2009). Techniques. In I. I. Analysis, *A Guide to the Business Analysis Body of Knowledger* (p. 102). International Institute of Business Analysis.

Laoyan, S. (2024, February 2). *What is Agile methodology? (A beginner’s guide).* Retrieved from Asana: https://asana.com/resources/agile-methodology

The University of Edinburgh. (2020, November 11). *Gibbs' Reflective Cycle*. Retrieved from The University of Edinburgh: https://www.ed.ac.uk/reflection/reflectors-toolkit/reflecting-on-experience/gibbs-reflective-cycle

*UnitTest++*. (2017, January 13). Retrieved from GitHub: https://github.com/unittest-cpp/unittest-cpp