SET10111- MultiAgent Systems

Assignment part 2 – Report

1. Overall Goal of the MAS

The overall goal of the Multi-Agent system is to simulate four colonies of ants that compete by searching for food and returning that food to the respective nest. Each ant is programmed as an intelligent agent that searches throughout its environment for food to be able to feed its colony. Each colony has five ants that search the environment separately. Interactions between ants of the same colony can be observer by the use of stigmergy. The ants use stigmergy to communicate and coordinate, when an ant is carrying food back its nest it drops a pheromone trail. Other ants of the same colony can detect this pheromone trail which notifies them that food has been found in that direction. This allows the ants of that colony to coordinate and ensure that the colony will be able to collect as much of the food from that location. This creates a fun visual simulation of multiple colonies of ants that are competing to attain the most food.

2. Features of the MAS

The Multi-Agent System developed for this project contains several features that enhance the virtual ant colony simulation.

Multiple Colonies
The simulation supports the use of multiple colonies of ants. This allows for the ants of differing colonies to compete with each other over resources contained in the environment.

Stigmergy
The ants communicate and collaborate with each other through stigmergy. This allows for the ants to communicate with each other by dropping pheromones when an ant has located a food item, telling the colony that food has been found at a certain location. The ants can then collaborate with each other, following the trail of pheromones and working together to gather as much food from that location.

Dynamic Environment
The environment is randomly generated, meaning it changes every simulation. The location of the nests is placed in a random location in a quadrant of the environment which means that some colonies may have a better placement than other depending on the simulation. The locations of food items are also randomly generated. At the beginning of the simulation a large number of food items are placed into random locations in the environment which could benefit some colonies more than others. Food is then spawned into the environment at tie intervals throughout the running of the simulation ensuring there is always food for the ants to find. The type of food spawned is based on weights that determine the chance the food has to spawn; less

valuable food is more likely to spawn, and higher value food items are rarer. This all makes the environment dynamic, always changing and ensures that no two simulations are identical.

Agent Behaviour
Each agent is capable of making their own decisions in regard to where they go and what they do. Every agent has a memory of the locations that they have visited and favour trying to explore undiscovered locations. Every agent's memory is independent from other ants in the colony, meaning that they explore the environment on their own. The agents have priorities that they follow in order to achieve the goal of collecting the most food as possible. These priorities can be simplified in the order of food first, pheromones second, end exploring third. The ants will always stop what they are doing when encountering food as this is what they want to collect. They will priorities the pheromones next as these pheromones are dropped when food has been found, Finally they prioritise exploration next, in order to locate new food for the colony.

Visualisation
The simulation employs the use of a visualisation of the Multi-Agent System so that the behaviours and interactions of the ant colony can be easily seen. The simulation uses PyGame to render the environment, ants, pheromone trails, nests, and food which allows the user to easily be able to understand what is happening within the Multi-Agent System. Each ant colony is represented by a distinct colour which is also shown on the respective nest. This colour is also used to be able to differentiate between the pheromone trails of the ant colonies. The ants are represented using images that rotate on the visualisation, showing the direction that the ant is heading in which makes it easier for the user to see what the ants are doing. Food items are differentiated in the environment using different images on the rendering of the visualisation, The size of the images is respective to the value of the food meaning that larger food items have a bigger value to the ant colonies. This visualisation makes it easier for users to comprehend what is going on with the simulation

User Interaction with the Visualisation
The user is able to interact with the visualisation of the simulation to be able to gain a better understanding of the Multi-Agent System. The simulation is able to be paused and un-paused, allowing the user to keep track of the actions of ants across the environment. When the simulation is paused the user can hover over items in the visualisation to gain a better understanding of what is happening at that moment of time. Hovering the mouse over the nests displays information on the colour of the nest and how much food has been deposited in the nest up until that point. The food items can be hovered over to display the name of the food item and also how much food is left at that location. The ants can also be hovered over, which displays the colour of the ant and the action that the ant is currently performing. These user interaction feature allow the user to gain insight into the behaviours of the agents and an understanding of the Multi-Agent System as a whole.

"Win" Condition
The simulation features a win condition that determines what ant colony performed

the best within that simulation. Once a certain number of food items have been deposited at a nest the simulation is stopped, and the winning ant colony is displayed to the user. This gives the simulation a point other than to just run endlessly and introduces an amount of competition between the ant colonies.

3. Description of Agent Oriented Design

Ant agent definition

Each ant agent an agent is autonomous and follows rules in order to explore the environment and bring as much food back to its nest as it can. The agents store a number of properties that describe the agents and its environment. The agents store the position and direction and the environment they are acting in to determine where they are in the environment. They store their colour and the nest they belong to in order to differentiate themselves from other colonies and to be able to know which pheromone trails they should follow. The sensing range of the agent controls how far the agent can sense, I chose to have it set to 2 which means the agent can sense two grid squares away from itself. This gives a good balance between the agents moving around and also fulfilling the goal of finding food. The agent maintains a list of its last fifty visited locations, the agents use this knowledge to try and explore new parts of the environment they have not been to yet. I chose this number as it forces agents to move in new directions in the short term, but it is short enough that its will not stop agents from returning to an already visited location.

The agents also a number of properties that are updated by the agent's actions. These parameters can then be used to affect how the agents act in the future. If an agent picks up a piece of food, it sets a Boolean carrying_food to true which can then affect the decision tree and change what the action of the agent will be. The following_pheromone property is set to True if the agent senses a pheromone trail, which tells makes the agent follow that branch of the decision tree. The ignore_pheromone_until property is set to a time in the future when an agent follow a trail to the end finding no food. This parameter stops the agent from following pheromone trails for a small amount of time to give a chance for the pheromone to fade away. This is in order to stop the agent from getting caught in a loop of following the pheromone, finding no food, exploring the environment, and then finding that same trail again. The sleep_until parameter is used to simulate the time it takes to complete certain actions and lets the agent just skip performing an action for a time. The agent_goal parameter is set when the agent is performing an action, It is only used for printing to the screen in the visualisation, but it reflects what decisions an agent has decided to do and makes it easier to understand the state of the agents. It is a string that explains the state of the agent, exploring, taking food to nest, picking food up, or following trail and is intended to give the users insight into what the agents are doing at a certain time.

Using these parameters the decision tree of the agents can be controlled to exhibit the intended behaviours. The code that controls the behaviour of the agents can be represented as a decision tree as seen below.

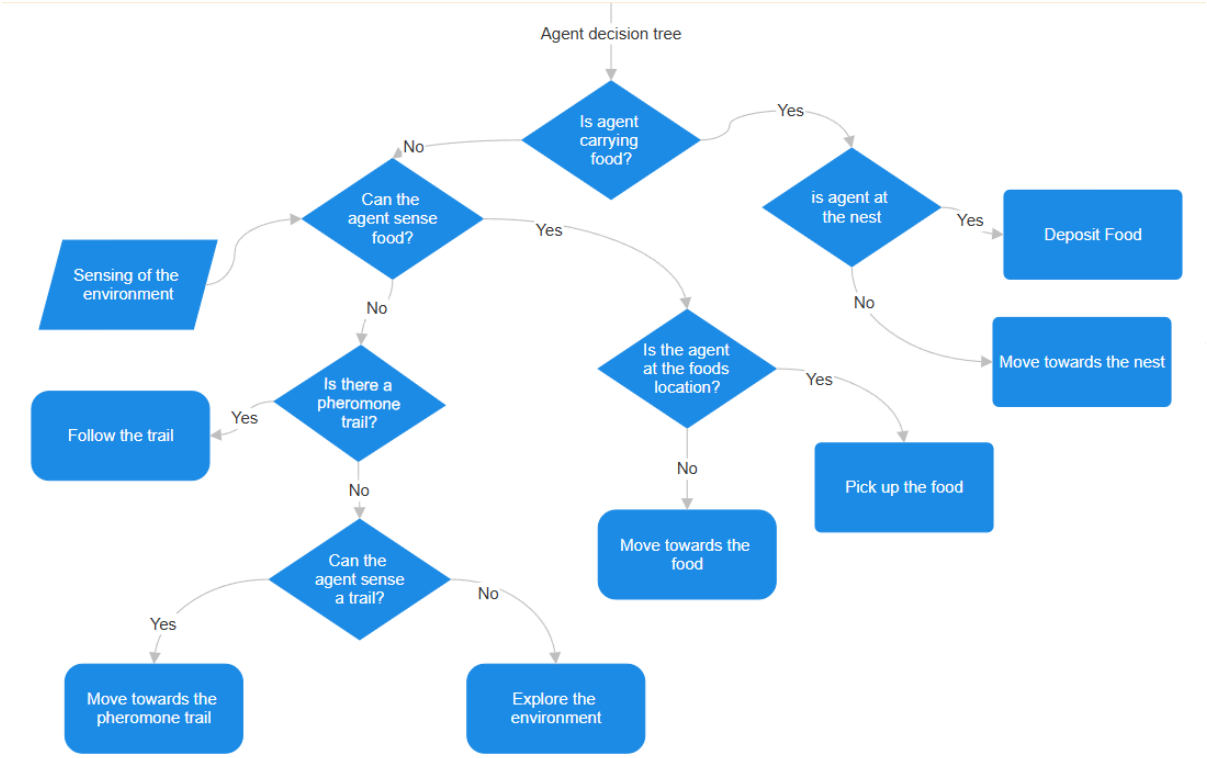

Figure 1: Decision tree of the ant agent

4. Features I Would Add if Given More Time

One feature of the system that I would add to extend the functionality of the Multi-Agent System is to introduce a more optimised search algorithm that better resembles the behaviour of ants in the real world. An algorithm such as the Ant Colony Optimisation algorithm could be used so that ants move around the environments in a more realistic and optimised way. This could even be extended to incorporate machine learning to improve the algorithm. The Ant Q algorithm uses reinforcement learning to enable agents to get better at searching and foraging for food the more the simulation is ran. This would allow the environment to become more complex, adding obstacles that cannot be traversed, the ants would find the optimal routes around these obstacles instead of moving directly to and from the nest.

Another feature I would include if I had more time is to a queen ant agent. This type of agent would reside in the nest and would keep track of what is happening in the environment through information communicated by the ant agents moving around the field when they return to the nest. The queen agent would be in charge of directing the other agents on where to go in the environment, ensuring that the ants search a larger area. This could keep track of multiple resource location and the location that the other ants have travelled to. When ants return to the nest the queen could use some sort of auctioning system to decide what the ant should be sent back out to do. For example, the queen knows two ants have been dispatched to collect a small food resource that was located earlier, the queen could then send the third ant

to explore a new area as it will not be efficient to send that ant to collect that resource. This could also be used to incorporate with the reinforcement learning mentioned earlier, queen ants of different colonies could then learn through their different resources and would display different behaviours.

The properties of the ants could also be differentiated between the different colonies. Properties such as movement speed, carry-weight, or sensory range could be tweaked so that there is some differentiation between the colonies other than location. This would make the simulation more competitive and the visualisation more entertaining.

Another feature I would have like to have implemented is heightened competition between the ant colonies. This would involve colonies being able to detect other colonies pheromones, allowing them to take the resources they have found in the environment or steal the collected resources from their nest. This could also integrate with the machine learning algorithms where maybe a certain colony realises that stealing collected resources is a more efficient source of food. Another type of agent could be introduced such as a soldier ant that has the job of protecting the nests from invading colonies. Another way of competition that could be introduced is to have some sort of battling system where and could fight each other to secure resources. Ants could do damage to each other and drive other colonies away from resources. This could be shown in the differing properties of ants, one colony is able to take more damage, or able to output more damage, a colony is more likely to forage in groups meaning more ants are able to do attack another ant. The soldier ant agents could also then be tasked to ensure the safety of foraging ants. All of these features add a myriad of events that could happen in the Multi-Agent System, increasing the depth of the simulation and the entertainment of the visualisation.

Finally, the last set of features I would like to add to the Multi-Agent System is a use for the resources that are collected. The way I envision this is to firstly add a lifecycle of the ants in the system. Ant eggs could be laid which uses resources, this could be controlled by the queen ant agent which decides if the colony has enough food for a new ant and if it currently needs another ant, The ants will also have a life expectancy which could be extended by consuming resources that have been collected. The ants would have to return to the nest, fostering greater opportunities for communication, and the queen ant agent would then have to balance to growth of the colony with the upkeep of the colony. This would also add greater depth to the simulation and ad to the entertainment value.

5. Reflection on challenges faced

One of the main challenges I faced was building upon the Multi-Agent System or introducing new features. Adding new features or changing how certain things worked was especially difficult due to how everything interacts within the system. Adding features were not as simple due to these added interactions as I had to think how the feature was going to function, how it would fit into the current system, and

how the feature would work with the interactions between the components. Such as adding more ant colonies, I had to change how the ants are initialise, I had to think about how the ants were going to interact with other pheromone trails, and how these differing trails were going to represented and handled by the environment which introduced a lot of backtracking and changing things during development for features that I thought were going to be relatively simple to implement. It was harder than usual to wrap my head around how new things were going to work, thinking up exciting new features was easy but implementing them were decidedly not. This led to many dead ends that had to be removed, a lot of changing how the already implemented features were coded, and many new features being reserved to being what-ifs. This was compounded upon by using the visualisation, which introduced another thing that I had to think about when introducing additional features. This highlighted the importance of planning when employing an agent-oriented approach to programming or designing a Multi-Agent System. Planning the capabilities of the system, what components will be used by the system, the types of agents, and how they interact with each other at the beginning of the project will make it much easier to implement the Multi-Agent System. Instead of trying to add new features and having to change things to make them fit, if I had thought out how features would work and how they would interact at the beginning it would have been less of a challenge to make the system more complex.

Another challenge I faced was implementing the visualisation. This kind of ties into the previous challenge slightly, where I had a simple console-based simulation that I got working first and tried to transfer over to a more complex graphical visual simulation. This took more effort than was expecting as I had somehow convinced myself it would be a case of just switching around some things around in the main function to rendering instead of printing them. This was not the case, and a lot had to be changed. This may have been due to my inexperience with how things are rendered and shown on the screen, it is all made to seem so simple when playing games or drawing on notepad, but I have now realised it is very much not. When designing applications, I usually tend to get the logic tested and working using a simpler console application then convert it to using a GUI, this approach was not as effective when dealing computer graphics and rendering. I should have planned for the visual simulation from the beginning and saved myself from the headache and extra time taken to switch everything over.