Swarm Intelligence for Autonomous Cooperative Agents in Battles for Real-Time Strategy Games

Damon Daylamani-Zad, Letitia B. Graham, Ioannis Th. Paraskevopoulos
Department of Computing and Information Systems
University of Greenwich
London, United Kingdom
{d.d.zad, gl213, i.parask}@greenwich.ac.uk

***Abstract*— this paper investigates the use the swarm intelligence of honey bees to create groups of co-operative AI for an RTS game in order to create and re-enact battle simulations. The behaviour of the agents are based on the foraging and defensive behaviours of honey bees, adapted to a human environment. The groups consist of multiple model-based reflex agents, with individual blackboards for working memory, with a colony level blackboard to mimic the foraging patterns. An agent architecture and environment is proposed that allows for creation of autonomous cooperative agents. The behaviour of agents is then evaluated and their intelligence is tested using an adaptation of Anytime Universal Intelligence Test.**

Keywords—Real-Time Strategy games; Swarm Intelligence; Agent Architecture; Multi-agent Intelligence;Artificial Intelligence; Finite State Machine

I. INTRODUCTION

Certain genres of video games require large numbers of individual artificial intelligence (AI) units to work together for or against the player, such as in Real Time Strategy (RTS) or Tower Defence games. Along with the ever increasing computer processing power, it is possible to use more complex algorithms and techniques to create more realistic and challenging AI with less computational expense. Multi-agent approaches and swarm intelligence are inspired on the ability of social animals and crowds to work together as a group without the need for a leader to delegate tasks. Individuals in a swarm are unable to find a solution to a colony's problems alone; however, by interacting with each other and making decisions based on local information, they can find a solution at the colony level [1].

The aim of swarm intelligence in AI is to create a decentralised group of autonomous, self-organised individuals that respond to local stimuli. When these individuals are viewed at the swarm-level, individual decisions should be contributing to the appearance of a group decision. Several algorithms are inspired by biological swarms; e.g., particle swarm optimisation is based on birds flocking [2] and ant colony optimisation is based on ant foraging methods [3].

The aim of this research is to adapt the real-life swarm intelligence of honey bees to create a group of co-ordinated AI agents for an RTS game setup. The goal is to recreate an autonomous AI that can be used to re-enact battles in a simulation format, which could also be used in history or military training.

This project will use elements of pre-existing algorithms, with the aim of developing an AI agent based behaviour of honey bees. This approach will not have a leader or brain to guide the strategy, instead the independent behaviour of agents aims to simulate an over-arching strategy. The behaviour and roles of honey bees are presented in section II and III, then existing approaches in swarm intelligence and multi-agent systems are presented. Section V presents instances of swarm intelligence being used in games. Section VI presents the proposed mapping of bees to soldiers, setting the ground for presenting the agent architecture and the proposed environment in section VII. The evaluation section shows the results of experiments and present an evaluation method based on Anytime Universal Intelligence Test. Finally the paper concludes in section IX.

II. BEHAVIOUR OF HONEY BEES

The next two section present the behaviour and roles within bee hives. These will be used as the basis of the agent behaviours and will be mapped to unit behaviour in an RTS in section VI. Group decisions in honey bee colonies are formed by lots of individual bees' decisions. According to Detrain and Deneubourg [4] an individual's decision can influence the decision of others, causing the appearance of a group decision. Individual decisions are made because of the bee's interactions with nest mates or the environment. Bees exhibit two main behaviours that is useful in a strategy battle games. These are Foraging and Defensive behaviour.

1) Foraging

When foraging, honey bees actively recruit others to food sources, providing information about the source through a waggle dance. Waggle dances consist of a series of waggle runs followed by a semi-circular turn; communicating the distance, the angular location based on the sun's azimuth and the odour of food. These dances provide bees with positive feedback that influences others to go to certain locations. The waggle dance is also used by bees to tell nest mates about suitable nest locations when a swarm is looking for a new home [5], [6].

While searching for the flowers bees will take an irregular path and possibly fly hundreds of meters from their hive; however, when they fly back to their nest after locating a food

source, they take the path with the shortest distance [7]. Experiments have shown that bees can learn and make decisions based on visual stimuli using their own working memory [8]. This working memory is what allows bees to navigate their environment and call on experience to make decisions about the profitability of food.

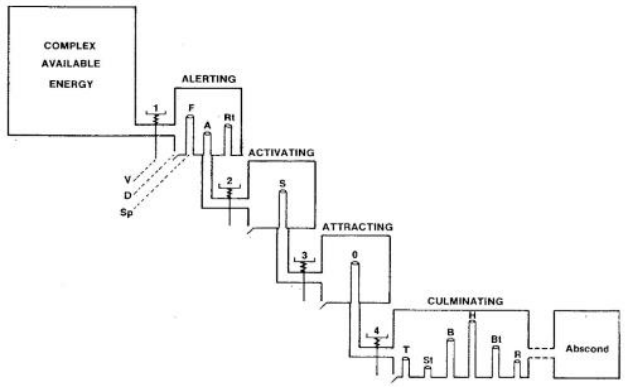

Fig. 1. Model of honey bee defensive behaviour [9].

2) Defensive Behaviour

The defensive behaviour of the colony can also be viewed as a collection of individual responses to stimuli from the environment, such as recruitment pheromones from colony members. Defensive responses of a bee can be broken down into four sequential phases: alerting, activating, attracting and culminating illustrated in Fig. 1. Repeated disturbances of the colony can invoke a fifth phase called absconding, whereby the queen and adult bees leave the nest [9].

In each of these phases there are number of actions a bee can take; however, they can only perform one action at a time and each action requires a certain amount of Complex Available Energy (CAE). By reacting to a stimulus, a bee can step through each of these phases and take an action. There are multiple forms of stimuli that can invoke defensive behaviour, including moving visual stimuli, vibrations of the nest and the alarm pheromones of nest mates [10].

a) Alerting

In the alerting phase, bees have the options of alert, recruit or flee.

- **Alert** bees take a defensive stance with their wings extended, mouth open and antennae waving. This response is not based on the direction of the stimulus, as it has been found that alert bees that are grouped together face in different directions.
- A **recruiting** bee will open their sting chamber and run into the hive, releasing alarm pheromone to stimulate nest mates into defensive behaviour. A recruiting bee can be recruited into further defensive action by their own pheromone.
- If the stimulus provides directional information, some bees will choose to **retreat** from the area of disturbance.

b) Activating

With more stimulation, bees reach the activating stage. Here, bees will look for the source of the disturbance. Depending on if the hive is open or unopened, the search will start close to the bee or the hive entrance. If there is no further stimulation after a period, the activated bee may begin searching meters from the hive.

c) Attracting

If an appropriate stimulus is found by an active bee, they will orient towards it. The same disturbance often simultaneously activates and attracts bees. This change in phase is more obvious when a stimulus is presented and then removed to a remote location.

d) Culminating

In this phase, several responses are possible. Bees may sequentially display two or more of the following responses: threaten, run, sting, bite, pull hair or burrow into clothes. If the integrity of the nest has been disrupted, the bee may choose to run.

III. Roles within a bee Colony

During a worker's lifetime, they will undertake different roles as they develop and age, also known as age-related polytheism [10]. The main roles they take are: Nurses, Middle Aged Bees, and Foragers.

Middle Aged Bees (MABs): develop around the age of 12 – 21 days and remain in at this stage of development for a little over a week. MABs can take on numerous roles within the nest. Younger MABs tend to take on tasks such as comb building and colony maintenance, while older MABs take on tasks closer to the hive entrance such as nectar receiving/processing and guarding [11]. Around 10 – 15% of MAB workers will take part in guarding [10]. The rest of the MABs tasks are closely related to foragers. Bees can choose which task to do based on feedback from the local environment, e.g. foragers performing the tremble dance can recruit MABs to act as receivers [4].

Foragers: Once a bee has developed into this role, they no longer take part in hive related tasks that MABs handle. Instead, foragers focus on collecting all resources [11].

There is a two more roles which will not be used in this paper. The **Cell Cleaner** role refers to newly emerged bees that have not been involved in the nest duties yet. The **Nurse** role is responsible for feeding the young and caring for the Queen.

IV. Multi-Agent & Swarm Intelligence

Swarm Intelligence is built upon the idea of multi-agent systems. A multi-agent system consists of an environment which multiple AI agents communicate and act within. An AI agent is described as something that can perceive its environment through sensors and make decisions/act based on information found in that environment [12]. An agent is expected to be able to:

- Operate autonomously
- Perceive their environment
- Persist over a prolonged period
- Adapt to change

- Create and pursue goals

Rationally, agents should aim to select an action that is expected to maximise its performance measure, based on evidence provided by the percept sequence and whatever built-in knowledge the agent has [12].

A. Model-based Agents

There are different agent types that can be created to replicate the behaviour of the honey bees. As the environment in which the bees are interacting with in a battle scenario is partially observable, sequential and dynamic, model-based reflex agents are the best fit for implementing the agents used within such environment [13].

These type of agents keep track of part of the world they cannot see anymore, by using information perceived historically. The state of the world the agent is tracking can be updated using information about how the world evolves independently of the agent, and how the agent's actions affect the world [14], [15].

Model-based reflex agents are most suitable for a partially observable environment, as they can maintain an internal state of the world that is dependent on the percept history. To be able to update this internal state, the agent needs to know how the world works; known as the model of the world. The model agents use needs two types of information: how the world evolves independently of the agent and how the agent's own actions affect the world. Using this state and the current percept, the agent can make decisions as to what to do.

As the environment is partially observable, the state maintained by the agent is better thought of as a best guess. This means there is likely to be uncertainty in the state, however decisions still need to be made [12], [15].

B. Blackboard architecture

A blackboard is a global structure that is available to all agents in a system to share information and collaborate to solve a problem. Traditional blackboard systems are made of three components: the blackboard, several knowledge sources (KSs) and a controller component. The blackboard acts as the shared memory for the KSs to read and write from. A KS is a system that can read information from the blackboard, process anything relevant to it, and contribute information towards solutions. KSs are independent of each other, and each can be a different type of system, allowing different approaches to problem solving; however, only one KS can write to the blackboard at a time. The controller component is responsible for choosing which KS can write to the blackboard next. Decisions are made based on what the KS will contribute and the resources required to create this contribution [16].

More recently, a different approach [17] was suggested for blackboards in a multi-agent system. Corkill proposed a system whereby each agent has their own blackboard and all KSs used in the system, allowing them to focus on nearby data and share with other agents they meet. This approach would create a flexible distributed system, similar to how honey bees use local information and their own experience to make decisions and share information. This has been taken on by other researchers and expanded to using a private and a shared blackboard, where part of the world is shared amongst the agents and other information are kept private to each individual agent [18].

C. Markov Decision Process for Decision Making

AI agents will need to be able to make decisions about the tasks they are performing, the information they share with others, the actions they will take during defensive behaviour and the nodes they will travel to, during resource gathering. There are many methods that can be used to allow agents to make these decisions. Due to the probabilistic nature of bees' foraging patterns and the model of defensive behaviour, the Markov Decision Process (MDP) is rendered to be the most suitable.

MDP is a process where all possible states are known, each state having related action and reward values, and a probability of transitioning from one state to another [19]. MDPs are most suitable to scenarios whereby the agent moves through several known sequential states, where transitions between states happen via a decided action. These points, where decisions are made, are known as decision epochs. Taking an action results in a reward value, which can be positive and negative; with negative values being a cost, rather than a reward. Agents know the rewards for actions before they are taken.

To handle a partially observable environment, Partially Observable MDPs (POMDPs) can be implemented. POMDPs have a similar structure to MDPs, however they require a sensor model to be able to create a belief state; a group of actual states the agent might currently be in [12].

V. APPROACHES IN GAMES

Multi-agent approaches including swarm intelligence have been used in games in recent years. These approaches have been used in both serious games used for military training, problem solving in battlefields, as well as used in traditional video games for entertainment.

Reynolds [20] proposed an approach in simulating the flocking behaviour of birds, creating a distributed behavioural model much like that at work in a natural flock as an alternative to scripting the path of each bird individually for animation.

Li and Hu [21] present a soccer simulation implemented using a multi-agent approach that implements a blackboard model so tackle the communication between the agents. Other researchers [22], [23], have used multi-agent blackboard approach to creating autonomous NPCs (Non-Playable Characters) that behave similar to humans in order to improve NPC behaviour and increase game engagement.

Particle Swarm Optimisation has been used in a tower defence game to optimise cannon locations to cause enemies the most damage [24]. The scenario contained two teams of players, attack and defence. The attack team needed to navigate along a single path that ensured minimum casualties. The defence team were given seven cannons to place on the map, with the aim of causing the maximum amount of damage to the enemy regardless of the path they took.

An adapted version of Ant Colony Optimisation (ACO) was used to simulate resource gathering in a real-time strategy

game environment that was believable to players. A memetic ant colony system was created, using ACO to explore the environment and communicate information about resources. This solution was chosen as it is less computational intensive than using explicit planning and searching. Experiments using this system with different levels of difficulty found that agents in this system were successful even when there were many obstacles and few resources available [25].

VI. Mapping the Bee Behaviour

The prior sections investigated the behaviour of honey bees that will be adapted for use in an RTS AI agent. This section discusses the findings and the proposed approach to implementing swarm intelligence into the agents within the game.

A bee, during its lifetime, will undertake several different roles. In the game environment, there will be several different roles that AI can undertake and switch between depending on the needs of the group e.g. Harvesters, Foragers, and Soldiers.

To replicate the bees working memory, each agent in the system will have their own private blackboard to store locally perceived information to be used for make individual decisions. To best use this working memory, model-based reflex agents will be implemented to keep track of the environment based on the models of honey bee behaviour discussed earlier.

The foraging behaviour is implemented into the game environment, using a shortest path calculation based on Dijkstra's algorithm to wander in the map and locate resources. Once resources are found the bees would move directly towards the base to deposit their load and will continue collecting using this new direct route. On returning to base, they will be able to access a global blackboard to which all agents can read and write. A successful forager can then add the location of the resource found to the blackboard, describing how to get to a resource in a similar way to the honey bee's waggle dance. The bees will use this point as a target direction for gathering resources. Negative feedback will occur when resources are depleted; the path is removed from the blackboard and other resources will need to be found.

If the nest is disturbed, units undertake defensive actions which are presented as the model for an individual bee. Bees step through the same phases of behaviour, however can take different actions in each phase dependant on how they perceive the disturbance. To handle the probabilities involved in choosing actions in defensive behaviour, a Markov decision process will be implemented. The process will allow agents to select different actions to move from phase to phase, dependant on the agent's remaining stamina. MDP could also be adapted to handle the decisions of which node to move to during foraging, dependant on the paths that have recently been travelled along.

VII. Environment and Roles

The simulation scenario involves a randomly generated environment that holds two bases (nests). This map will host two separate colonies known as the Defenders and the Aggressors. There are resources scattered around the environment which the Defenders aim to collect as their base would produce a new agent after each ρ amount of resources collected. Aggressors would be scouring the environment, hunting Defender agents and searching for the Defender's base. This environment is created inspired by Λ environment [26], [27] which is one of the environment classes that implements the theory behind the Anytime Universal Intelligence Test [28].

Once the aggressors have spotted the defender's base they will stop foraging and gather to attack. As the same time, as the aggressors have also been spotted, the defenders would issue a "Call to Arms" (alert->recruiting) which would result in recruiting the defenders into a defensive position ready for the imminent attack from the aggressors. The simulation will arrive at its conclusion when both sides line up and charge each other resulting in battle. Once a side is completely wiped out the simulation ends.

If the nest is disturbed, units undertake defensive actions which are presented as the model for an individual bee. Bees step through the same phases of behaviour, however can take different actions in each phase dependant on how they perceive the disturbance. To handle the probabilities involved in choosing actions in defensive behaviour, a Markov decision process will be implemented. The process will allow agents to select different actions to move from phase to phase, dependant on the agent's remaining stamina. MDP could also be adapted to handle the decisions of which node to move to during foraging, dependant on the paths that have recently been travelled along.

A. Defender roles

- Harvesters (Foragers): collect resources only, cannot attack but can alert the castle about attackers. The have Low CAE.
- Militia (MAB): collects resources, but can switch to defensive mode to protect castle when called.

B. Aggressor roles

- Scouts (MAB): do not collect resources, only looking for defenders or the castle, can attack when called to arms, can travel further than Attackers and can detect enemies/castle in a wider range.
- Attackers (MAB): do not collect resources, can look for defenders/castle, can attack but cannot travel as far as Scouts unless attacking a castle.

C. Resources

The resources within the map are used for replenishing the health and the energy of defenders. The collected resources will also allow the defender's castle to build new defender units. The aggressor's start the game with existing resources presumed to have been gained from previous raids.

D. Agent Architecture

The proposed architecture for the agents in presented in Fig. 2. The agent consists of blackboard and processing components. Information is received by the sensor form the environment, this information is then stored in the blackboard. The Action Selector unit would than select a suitable action

based on the information on the blackboard. This final decision is then passed on to the actuator which would apply this to the environment, hence exhibiting a behaviour.

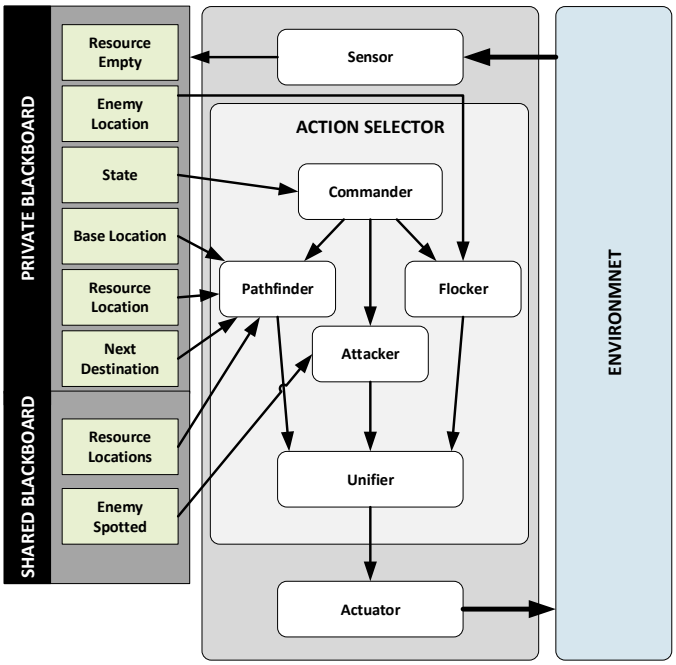

Fig. 2. Agent architecture

The blackboard is divided into two sections; a private blackboard and a shared blackboard. The private blackboard is private and independent for each agent. The information on the private blackboard are only accessible to an instance of the agent and relate to this agent. The shared blackboard is information that is shared amongst all agents. This simulates the knowledge held at the nest that is publicly known to all agents.

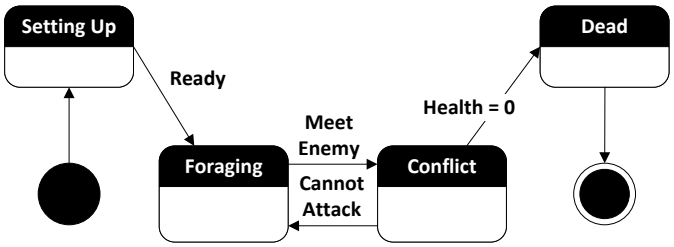

Fig. 3. High level agent state machine.

The action consists of a commander unit that reads the current state of the agents and then based on the current state of the world decides which of the underlying experts are needed to make the next decision. At any given situation, the agent might be needing to take multiple decision through multiple experts. The agent might need to move and flock at the same time as flocking action would involve moving and therefore pathfinding. The three expert units; pathfinder, attacker and flocker will use the information from the agent state and the world state and make their own relative decisions. Finally the decisions made by the three experts are passed to the unifier unit which would form the decision into actionable directives that are passed to the actuator.

The commander and the three expert units base their decision making on a hierarchical state machine of the agents that represents the states which the agents would go through based on the information received from the game as well as the decisions that the agents has taken. Fig. 3 represents the high level agent state machine. Each agent after it is created would go through a setting up state where its role and initial action is decided. Once the agent is ready, it will go to the foraging state where as the aggressor or defender it will start to look for defenders or resources respectively. From foraging, if they meet enemies, they will go to attacking or defending states and finally when an agent's health reaches zero it will die.

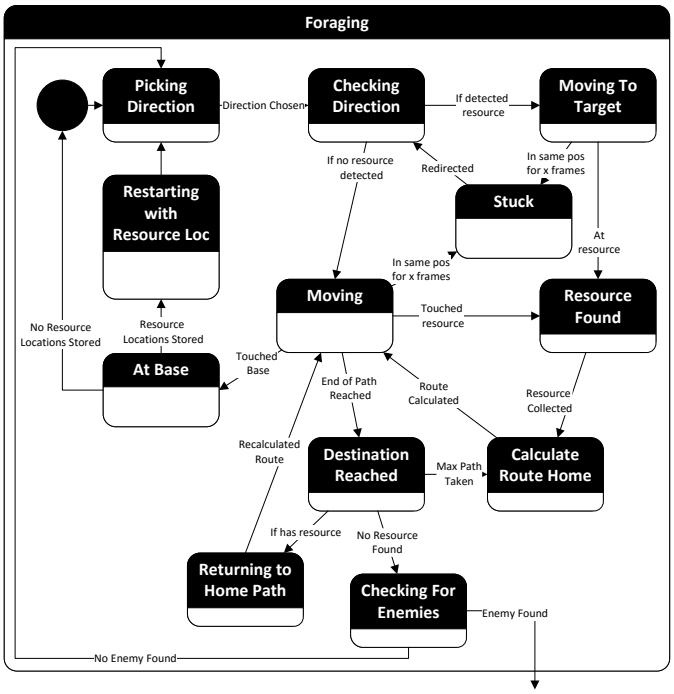

Fig. 4. Foraging state

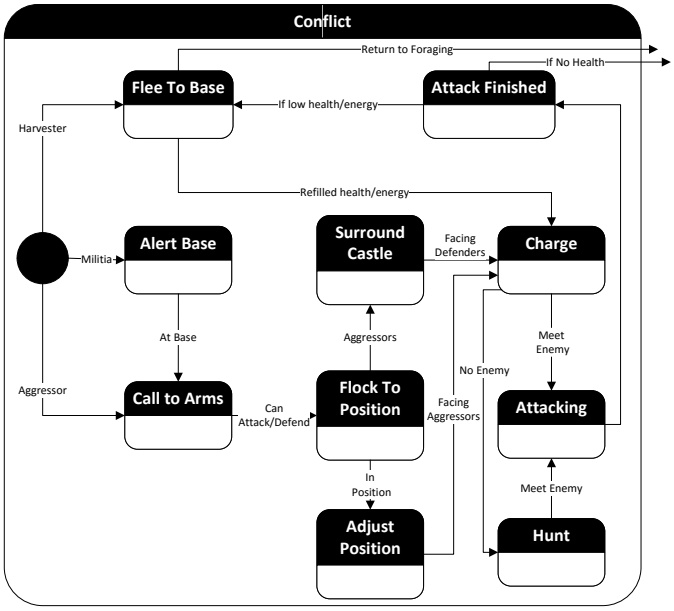

Fig. 5. Conflict State

The foraging state, illustrated in Fig. 4, encompasses the state machine for movements and path finding. This state machine would allow foraging unless an enemy or resource has been spotted in which case the agent would move to the conflict state or focus of collecting resources.

As presented in Fig. 5, conflict state controls the decision for the various agent roles during the disturbances. The aggressor agents will call to arms all their fellow attackers and scouts, flock to position and charge the defenders. The defenders on the other side will get into formation, and charge the aggressors to defend their castle. There is a chance that the harvesters might flee to base instead of joining the defence. The aggressors have a special state, Hunt, as they would hunt for enemies if they are not attacking the castle.

TABLE I. MAPPING BEE DEFENSIVE BEHAVIOUR TO UNIT BEHAVIOUR.

Behaviour			**Cost (CAE)**	**Damage**
Bee	***Human***			
	Aggressor	***Defender***		
Recruit	Initiate Call to Arms	Flee	-	-
	Call to Arms at Base	Call to Arms at Base	-	-
Searching	Get in Formation	Get into Defensive position	-	-
Attract	Get in line for charge	Get in line for charge	-	-
Culminating	Battle Cry	Taunt	1	+1 to the next attack
	Sword Attack	Sword Attack	1	1
	Block with Shield	Block with Shield	2	No Damage
	Shield Bash	Shield Bash	5	2
	String Sword Attack	String Sword Attack	8	4
	Dodge	Dodge	3	No Damage

The table 1, illustrates how the bee behaviour model presented earlier has been mapped to human behaviour during disturbances to the nest. The table shows how recruiting, searching, alerting, attract and culminating will be mapped to simulated human behaviour in the AI units within the game as well presenting the mapping of attack patterns and their weight in the units for the MDP to decide on actions.

If the nest is disturbed, units undertake defensive actions which are presented as the model for an individual bee. Bees step through the same phases of behaviour, however can take different actions depending on the CAE levels and their role. The agents would use the table below to decide also on the rewards of the actions they take compared to the cost.

The flock to position state deals with the flocking behaviour of the agents. This state machines, presented in Fig. 6, allows the agents to make decision on their positioning, when responding to call to arms, or when readying to get in line for charge. The flocking behaviour is at the core of crowd and swarm intelligence. The flocking patterns are a great demonstration of independent agents behaving separately and yet creating a crowd pattern.

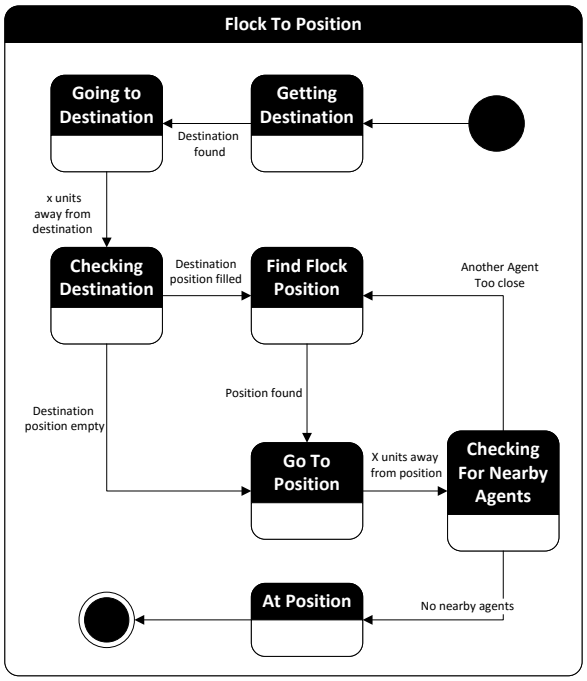

Fig. 6. Flock to Position state

VIII. EVALUATION

This section presents the evaluation of the proposed architecture. The evaluation method used here is inspired by [29] and their approach to Anytime Universal Intelligence Test [28]. For this purpose the environment is created based on a Λ environment [26], [27]. The idea is to evaluate an agent that can perform a set of finite tasks based on the environment it is placed in. The intelligence of the system is assessed as a function of successful transition between states and the success of achieving the objective of each state. Fig. 7 illustrates the agents in three different states.

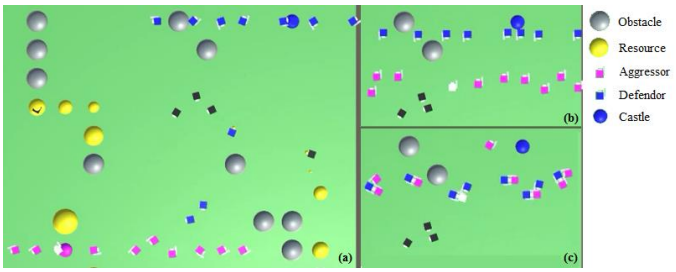

Fig. 7. Simulation of the game. (a) Flock to Position at base. (b) Adjusting Position to Charge. (c) Charge.

Each agent has its own specific role as defined earlier. Each of these roles would have their own specific set of tasks and states that they are supposed to achieve. Each agent π_i which is a member of $\Pi = \{\pi_1, \pi_{2\ldots} \pi_n\}$ has a role ω_j from $\Omega = \{\omega_1, \omega_2 \ldots \omega_m\}$. Each role would have a set of states available to them which is defined as $S_j = \{s: s \in States \wedge \omega_j \text{ can be in } s\}$, where *States* is the set including all the states available.

An available state s_k would have a good outcome $\oplus_k$ and a bad outcome $\otimes_k$. The good outcome is achieving its objective whilst the worst outcome is the complete opposite. This is a

theoretical definition that has been put into practice based on the tasks at hand. For example for a harvester agent which is in found resource state, the good outcome would be to collect resource and go home. But the bad outcome in this scenario has been defined as not going home i.e. being stuck or moving towards any other position, either case, it would be the destination chosen by the agent as opposed to the expected destination.

A reward function has been defined for the agents which would represent how an agent is performing in a state based on the best and worst outcomes of that state. The reward of agent π_i in state s_k is represented by $\gamma_{i,k}$ where $-1.0 \leq \gamma \leq +1.0$. The value of $\gamma_{i,k}$ is calculated using (1) as a function of the outcome of the state objective.

$$\gamma_{i,k} = 1/f(\pi_i , \oplus_k) - 1/f(\pi_i , \otimes_k) \quad (1)$$

Where f(a, b) denotes the success of a achieving b. this can take many forms as in previous example it would be the distance of agent a from the position b. other examples could include in case of a conflict f(a, b) would take the form of successful implementation of attack b by agent a.

The intelligence of an agent π_i would be defined as the amalgamation of its rewards in all its states during an iteration of the simulation, denoted as I_i presented in (2). Hence the intelligence of the agents in a role, $\omega_j I$, can then be calculated using (3) as the amalgamation of all the agents that are acting in the role ω_j.

$$I_i = \sum_{k=1\ldots p} (\gamma_{i,k}) / p \quad (2)$$

$$\omega_j I = \sum_{i=1\ldots q} I_i / q \quad (3)$$

The number p in (2) is the number of states available for agent π_i and it is important to note that if an agent arrives at and therefore p would not necessarily equal to $|S_i|$. It would depend on the states that the agent has arrived at during a simulation which means some states might be skipped whilst others might have been repeated many times. The number q represents the number of π_i agents that are acting in role ω_j.

Finally, the collective intelligence of a set of agents, Π, is defined as $\psi(\Pi)$ and calculated using (4) as the amalgamation of the intelligence of all the roles.

$$\psi(\Pi) = \sum_{j=1\ldots m} \omega_j I / m , m = |\Omega| \quad (4)$$

As Ω is the set of all the roles and each agent within Π is mapped to a role, by association, amalgamating the intelligence of each role, would allow for an amalgamation of the intelligence of all agents.

For the purpose of this research, the architecture was implemented using C# and Unity and the simulation was executed for five iterations. Each iteration concluded with one side, either aggressors or defenders, defeated and wiped out. The iteration took between 6-11 minutes and both sides had an equal number of agents spawned at the start. A sample iteration can be viewed here[1]. Table II presents the amalgamated rewards for each state for the five test iterations.

[1] https://youtu.be/VX8QUYdpqsg

TABLE II. AMALGAMATED REWARD FOR EACH STATE.

State	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5
Adjust Position	0.978	0.97	0.88	0.8	0.88
Alert Base	0.94	0.94	0.75	0.99	0.8
At Home	0.89	0.86	0.83	0.97	0.782
At Position	0.98	0.74	0.79	0.71	0.77
Attack	0.908	0.91	0.93	0.85	0.972
Attack Finished	0.838	0.81	0.89	0.96	0.881
Call To Arms	0.79	0.87	0.93	0.84	0.98
Charge	0.698	0.95	0.71	0.94	0.936
Checking Dest. Pos.	0.84	0.76	0.81	0.9	0.827
Checking Direction	0.86	0.82	0.95	0.8	0.943
Checking For Enemies	0.8	0.98	0.94	0.7	0.991
Default	0.90	0.95	0.90	0.95	0.985
Find Flock Position	0.96	0.85	0.75	0.73	0.789
Flee To Base	0.71	0.96	0.71	0.95	0.869
Found Resource	0.95	0.75	0.99	0.74	0.79
Getting Destination	0.84	0.74	0.87	0.83	0.947
Get To Position	0.86	0.78	0.99	0.86	0.9
Going Home	0.73	0.9	0.87	0.7	0.891
Going To Dest. Pos.	0.8	0.77	0.89	0.82	0.967
Go To Flock Position	0.89	0.98	0.94	0.86	0.838
Hunt	0.93	0.777	0.936	0.94	0.736
Moving	0.88	0.75	1	0.89	0.825
Moving To Target	0.81	0.9	0.95	0.87	0.81
Pick Direction With Target	0.99	0.87	0.86	0.82	0.71
Picking Direction	0.81	0.83	0.85	0.8	0.991
Restarting With Resource Loc.	0.86	0.71	0.9	0.87	0.78
Returning To Path Home	0.91	0.95	0.79	0.94	0.82
Starting	0.97	0.89	0.77	0.74	0.772
Stuck	0.727	0.91	1	0.82	0.707
Surround Castle	0.92	0.72	0.75	0.93	0.9
Adjust Position	0.78	0.97	0.88	0.7	0.88

TABLE III. INTELLIGENCE FOR EACH ROLE.

Role	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5
Attacker	0.94	0.89	0.86	0.81	0.86
Scout	0.9	0.92	0.77	0.84	0.87
Militia	0.8	0.922	0.977	0.83	0.758
Harvester	0.9	0.7	0.94	0.93	0.74

TABLE IV. COLLECTIVE INTELLIGENCE FOR EACH TEAM.

Team	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5
Aggressors	0.92	0.905	0.815	0.825	0.865
Defenders	0.85	0.811	0.958	0.88	0.749

Tables III and IV respectively represent the intelligence calculated for each role and then the collective intelligence of each team. The intelligence values are generally above 0.7, this is considered a very high score as the range of possible intelligence would have been between -1 and +1. The intelligence scores are close to the top end meaning that the majority of times the agents have been consistently pursuing the good outcome for each state. Whilst on average it seems to be the aggressors are performing slightly better that the defenders, it is worth noting that there are iterations such as iteration 3 and 4; where defenders have performed better.

IX. CONCLUSION

This research investigated the idea of adapting swarm intelligence honey bees when foraging and defending their nests to create a group of co-operative agents. The idea was to create a decentralised group of autonomous agents that could work together to achieve goals that would be found within an RTS game. The goal was to simulate battle behaviour without a central control or a leader that would dictate strategy [30], [31].

Instead, trying to create the battle out of a swarm behaviour of independent agents. Multiple simulations were run and using the recorded data, patterns of behaviour were identified and analysed to check whether the performed behaviour was expected for that role and main state. The analysed data was then used to calculate the intelligence score of each individual agent using the Λ environment and the Anytime Universal Intelligence Test. The agents scored consistently above 0.7 in a scale between -1 and +1, demonstrating a high level of collective intelligence. A noteworthy behaviour also emerged, presenting the appearance of team leaders during the setup of a larger battle. This behaviour goes against the idea of creating a decentralised group, however due to the nature of an RTS environment, the temporary appearance of leaders makes sense to lead the teams together into battle.

Future work for this project revolves around scaling up the simulation in terms of environment, roles available in the teams and the action taken during battles. To introduce new behaviours successfully, the environment will need to be larger to handle more agents and more detail. This would allow the system to be expanded and applied to other game genres such as shooter games, racing games and especially serious games and simulations such as rehabilitation games and narrative based trainings. Also, introducing a new role in the defender team called the Scavenger which would act in a similar way to harvesters, yet would take part in battles. They would be able to collect resources from the corpses of dead agents, providing another resource for creating defender agents. Currently, agents can fight on a 1vs1 basis, when an enemy is nearby or in a larger battle after spotting the defender base. In a larger environment, it would be possible to call groups of nearby agents together for group battles to occur. If deciding to call other agents for a group battle, agents within a certain radius would create formations and attack each other. Hence brining the simulation much closer to true strategic behaviour and battle simulations.

REFERENCES

[1] S. Garnier, J. Gautrais, and G. Theraulaz, "The biological principles of swarm intelligence," *Swarm Intell.*, vol. 1, no. 1, pp. 3–31, 2007.

[2] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, 1995, vol. 4, pp. 1942–1948.

[3] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey," *Theor. Comput. Sci.*, vol. 344, no. 2–3, pp. 243–278, Nov. 2005.

[4] C. Detrain and J.-L. Deneubourg, "Collective Decision-Making and Foraging Patterns in Ants and Honeybees," in *Advances in Insect Physiology*, Alexander S. Raikhel, Ed. Academic Press, 2008, p. (35) 123-173.

[5] R. Menzel, "Learning, memory, and 'cognition' in honey bees," *Neurobiol. Comp. Cogn.*, pp. 237–292, 1990.

[6] T. D. Seeley, P. K. Visscher, and K. M. Passino, "Group decision making in honey bee swarms," *Am. Sci.*, vol. 94, no. 3, p. 220, 2006.

[7] K. von Frisch, *Tanzsprache und Orientierung der Bienen*. Berlin Heidelberg: Springer-Verlag, 1965.

[8] S. Zhang, F. Bock, A. Si, J. Tautz, and M. V Srinivasan, "Visual working memory in decision making by honey bees," *Proc. Natl. Acad. Sci. United States Am.* , vol. 102, no. 14, pp. 5250–5255, Apr. 2005.

[9] A. M. Collins, T. E. Rinderer, K. W. Tucker, H. A. Sylvester, and J. J. Lackett, "A Model of Honeybee Defensive Behaviour," *J. Apic. Res.*, vol. 19, no. 4, pp. 224–231, Jan. 1980.

[10] G. J. Hunt, "Flight and fight: A comparative view of the neurophysiology and genetics of honey bee defensive behavior," *J. Insect Physiol.*, vol. 53, no. 5, pp. 399–410, May 2007.

[11] B. R. Johnson, "Division of labor in honeybees: form, function, and proximate mechanisms," *Behav. Ecol. Sociobiol.*, vol. 64, no. 3, pp. 305–316, 2010.

[12] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach, 3rd edition*. Upper Saddle River, NJ, USA: Prentice Hall, 2009.

[13] R. Lieck and M. Toussaint, "Temporally extended features in model-based reinforcement learning with partial observability," *Neurocomputing*, vol. 192, pp. 49–60, Jun. 2016.

[14] D. H. Scheidt and M. J. Pekala, "Model-based agents," in *Power Engineering Society General Meeting, 2007. IEEE*, 2007, pp. 1–2.

[15] K. J. Miller, C. D. Brody, and M. M. Botvinick, "Identifying Model-Based and Model-Free Patterns in Behavior on Multi-Step Tasks," *bioRxiv*, Dec. 2016.

[16] D. D. Corkill, "Blackboard systems," *AI Expert*, vol. 6, no. 9, pp. 40–47, 1991.

[17] D. D. Corkill, "Collaborating Software: Blackboard and Multi-Agent Systems & the Future," in *Proceedings of the International Lisp Conference*, 2003.

[18] J. Orkin and D. Roy, "The restaurant game: Learning social behavior and language from thousands of players online," *J. Game Dev.*, vol. 3, no. 1, pp. 39–60, 2007.

[19] Mausam and A. Kolobov, "Planning with Markov Decision Processes: An AI Perspective," *Synth. Lect. Artif. Intell. Mach. Learn.*, vol. 6, no. 1, pp. 1–210, Jun. 2012.

[20] C. W. Reynolds, "Flocks, Herds and Schools: A Distributed Behavioral Model," in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, 1987, pp. 25–34.

[21] S. Li and F. Hu, "Communication between the RoboCup Agents Based on the Blackboard Model and Observer Pattern," in *Proceedings of the 5th International Conference on Wireless Communications, Networking and Mobile Computing*, 2009, pp. 1–5.

[22] J. Orkin and D. Roy, "Automatic Learning and Generation of Social Behavior from Collective Human Gameplay," *Proc. 8th Int. Conf. Auton. Agents Multiagent Syst.*, vol. 1, pp. 385–392, 2009.

[23] J. Orkin, "Using online games to capture, generate, and understand natural language," in *the 13th European Workshop on Natural Language Generation*, 2011, p. 71.

[24] P. Huo, S. C. K. Shiu, H. Wang, and B. Niu, "Application and Comparison of Particle Swarm Optimization and Genetic Algorithm in Strategy Defense Game," in *Proceedings of Fifth International Conference on Natural Computation*, 2009, pp. 387–392.

[25] X. Chen, Y. S. Ong, L. Feng, M. H. Lim, C. Chen, and C. S. Ho, "Towards Believable Resource Gathering Behaviours in Real-time Strategy Games with a Memetic Ant Colony System," *Procedia Comput. Sci.*, vol. 24, pp. 143–151, 2013.

[26] J. Insa-Cabrera, J. Hernández-Orallo, D. L. Dowe, S. Espana, and M. V. Hernández-Lloreda, "The anYnt project intelligence test: Lambda-one," in *Proceedings of AISB/IACAP 2012 Symposium "Revisiting Turing and his Test: Comprehensiveness, Qualia, and the Real World,"* 2012, pp. 20–27.

[27] N. Chmait, D. L. Dowe, Y.-F. Li, D. G. Green, and J. Insa-Cabrera, "Factors of collective intelligence: How smart are agent collectives?," in *Proceedings of 22nd European Conference on Artificial Intelligence (ECAI)*, 2016, pp. 542–550.

[28] J. Hernández-Orallo and D. L. Dowe, "Measuring universal intelligence: Towards an anytime intelligence test," *Artif. Intell.*, vol. 174, no. 18, pp. 1508–1539, 2010.

[29] N. Chmait, Y.-F. Li, D. L. Dowe, and D. G. Green, "A Dynamic Intelligence Test Framework for Evaluating AI Agents," in *Proceedings of Evaluating General-Purpose AI (EGPAI), ECAI workshop*, 2016.

[30] I. Karpov, L. Johnson, V. Valsalam, and R. Miikkulainen, "Evaluation Methods for Active Human-Guided Neuroevolution in Games," in *2012 AAAI Fall Symposium on Robots Learning Interactively from Human Teachers (RLIHT)*, 2012.

[31] I. V Karpov, L. M. Johnson, and R. Miikkulainen, "Evaluating team behaviors constructed with human-guided machine learning," in *Proceedings of the IEEE Conference on Computational Intelligence in Games*, 2015.