ATHABASCA UNIVERSITY

MULTI-AGENT SYSTEMS (MAS) MODELLING ENVIRONMENT FOR HUNTING AND EVASION ALGORITHMS

BY

MARC PRINCE

A project submitted in partial fulfillment
Of the requirements for the degree of
MASTER OF SCIENCE in INFORMATION SYSTEMS

Athabasca, Alberta

September 2017

© Marc Prince, 2017

DEDICATION

This research is dedicated to the NetLogo Community which may assist future generations of researchers in evolving our understanding of Autonomous Agents and Multi-Agent Systems.

ABSTRACT

While new hunting algorithms are being developed to model pack animal hunting behaviours, these are being assessed with statistical methods rather than with simulation methods. The deficiency with statistical methods is that they represent a centralized approach to evaluating hunting algorithms rather than the decentralized method that exists in reality with real hunting agents (such as wolves and lionesses). Put another way, there is no central agent that controls the behaviour of real hunters (they independently collaborate towards common goals), thus any attempt to simulate these through a centrally controlled algorithm is artificial. To address this deficiency, this research proposed, designed, and developed a Hunting Algorithm Performance Evaluation Environment (HAPEE) using Multi-Agent System (MAS). Based on the Belief, Desire, and Intent architecture, the MAS were developed with the Netlogo agent-based modelling environment which includes the World Agent. The World Agent was customized to a limited number of two-dimensional scenarios with both Hunting Agents (HA) and Prey Agents (PA) interacting with each other and with the obstacles within the scenarios. The Hunting Agents and Prey Agents were developed with the flexibility to instantiate many types of hunters and prey. The differences in both the HAs and PAs was with their skills (communications, perception, speed, etc.) and their cognitive abilities. Finally, the HAPEE was used to evaluate two hunting algorithms: the Lion Optimization Algorithm (LOA) and the Grey Wolf Optimazation (GWO) algorithm. The experimental results show that the LOA is more resilient to obstacles than is the GWO. In the presence of obstacles, the lionesses were more reliable in completing joint convergence onto the prey (with the Alpha attacking first followed by the Beta). While the wolves had a lower convergence rate, they displayed an ability to recover from the confusion caused by obstacles to finally complete convergence (though it took much longer). Thus the lionesses either succeeded or failed to converge, whereas the wolves could initially fail

to converge, but then re-engage and succeed to converge at a later attempt. So while the lionesses were fairly binary in success or failure to converge on the prey, the wolves displayed a broader tolerance to recover from failure to finally converge onto the prey.

PREFACE

This research was conducted to determine the feasibility of developing a comprehensive hunting environment within which to test search, hunt, and evasion algorithms. The NetLogo Agent-Based-Modelling environment was chosen as it is open-source and very widely used by the academic community. By restructuring the NetLogo code into the Belief, Desire, Intent (BDI) framework, it is hoped that this research will enable future academics to develop increasingly complex hunting environments and help learn and test hunting algorithms.

This thesis paper should be read in conjunction with the NetLogo model which was programmed using NetLogo version 6.0. The NetLogo model can be downloaded from the NetLogo Modelling Commons at http://modelingcommons.org/account/login under the title of Hunting Algorithm Performance Evaluation Environment (HAPEE).

LIST OF TABLES

Page

1. Procedures Summary for the HAPEE..... 30

2. Summary of LOA evaluation results..... 31

3. Summary of GWO evaluation results.....31

LIST OF SYMBOLS, ABBREVIATIONS AND NOMENCLATURE

AA	Autonomous Agent
BDI	Belief, Desire, Intent
GA	Generic Agent
GWO	Grey Wolf Optimization
HA	Hunting Agent
HAPEE	Hunting Algorithm Performance Evaluation Environment
LOA	Lion Optimization Algorithm
MAS	Multi-Agent Systems
NetLogo	MAS development environment
PA	Prey Agent
WA	World Agent

with the environment itself (i.e., the World Agent). The World Agent would be a dynamic environment that can be adapted to represent the conditions that exist in real world hunting environments. The HAPEE would be programmed in NetLogo and thus provide the tools to evaluate the performance of search, hunting, and evasion algorithms. The hunting algorithms will consist of the Grey Wolf Optimizer algorithm (Mirjalili, Mirjalili, & Lewis, 2014) and the Lion Optimizer Algorithm (Yazdani & Jolai, 2015). All three of the agents (WA, PA, and HA) will be structured as closely as possible to the Belief-Desire-Intent (BDI) architecture as detailed in Intelligent Agents (Wooldridge, 1999). The completed HAPEE model would then be submitted to the Netlogo development group for inclusion in the Models Library.

Hypothesis

The hypothesis postulates that NetLogo MAS Development Environment, structured in a modified BDI framework, provides a feasible MAS Hunting Algorithm Performance Evaluation Environment thus providing an alternative means to test Hunting Algorithms.

Organization of the Remaining Chapters

The remaining chapters are organized as follows. Chapter II reviews the existing literature and current related research. Chapter III describes the methodology of implementing the simulation environment with NetLogo and using it to evaluate the effectiveness of the various search, hunt, and evasion algorithms. Chapter IV examines the test results of the Grey Wolf Optimizer algorithm and the Lion Optimizer Algorithm. Chapter V discusses the findings of the research and the results of the algorithm testing. The last chapter concludes the research effort and proposes recommendations for further research within this field.

CHAPTER II

REVIEW OF RELATED LITERATURE

Related Work

The related work in this field can be divided into three main groups as displayed in the Concept Map of the Literature in Figure 1 below:

a. Autonomous Agent Design Theory (denoted by Green Bubbles).

b. Hunting and Prey Behavioural Algorithms (denoted by Blue Bubbles).

c. Autonomous Agent and MAS Design Environment (denoted by Black Bubbles).

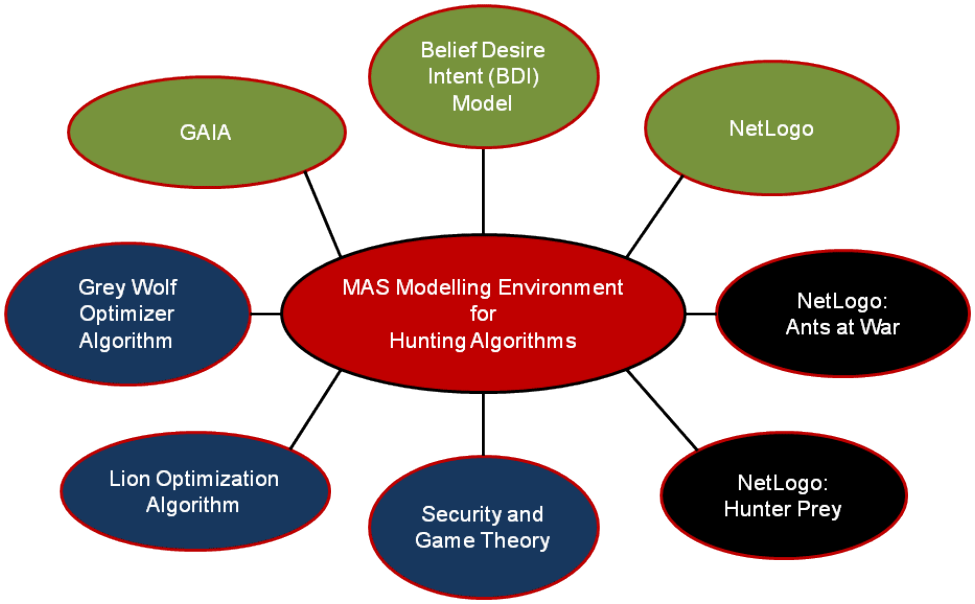

Figure 1: Concept Map of the Literature

The various areas of related work are described in more detail in the following sub-sections.

Autonomous Agent Design Theory and Architectures. While many design models exist for Autonomous Agents, the three considered for this research were Belief-Desire-Intent (BDI), GAIA, and NetLogo.

environment. AgentSpeak [Machado & Bordini, 2003] is a design environment built specifically for a BDI architecture which has limited World Agents (these are custom built to a specific MAS implementation). V-Rep [Coppelia Robotics, 2017] is a design environment based on existing, commercially available robots which is not structured on any particular AA architecture. It has a three-dimensional World Agent which controls a limited surface area. Because it runs in 3 dimensions, it is computationally demanding and can support only a limited number of simultaneous Autonomous Agents. Though NetLogo has its native MAS architecture [Wilensky & Rand, 2015], it appears to be architecturally agnostic as seen by Wilhelmy, R. et al. (2014) who successfully adapted it to the GAIA architecture. It has a dynamic World Agent which can be programmed in two (2) and three (3) dimensions. It can also be interfaced with the Arduino robotic platforms to extend the AA and MAS into the physical world.

Search, Hunting and Evasion Algorithms. Many search pattern algorithms exist with respect to searching for objects or for prey. These patterns can be for an individual to execute by themselves or for groups to execute in a collaborative effort. Examples of search patterns include the expanding square search pattern, the expanding series of concentric circles pattern, and the track line pattern (see Figure 2 below).

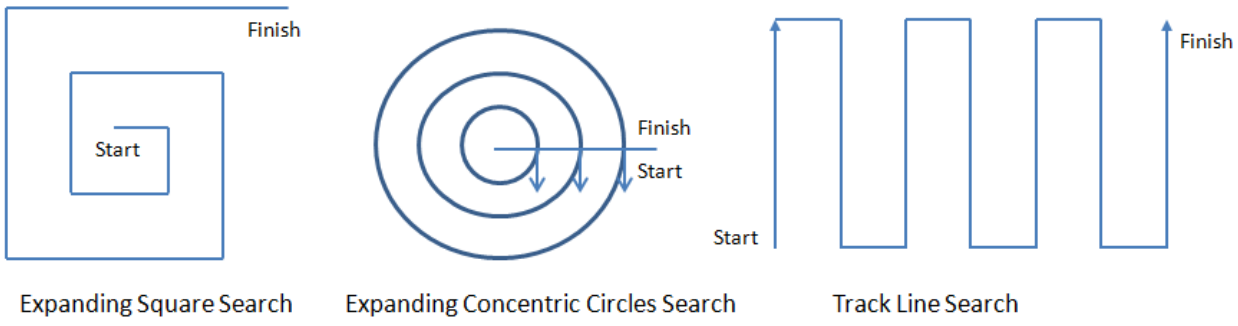

Figure 2: Search Patterns

Much literature exists on the hunting techniques of animals and many of these have been captured in videos and are available in television programming and most recently on YouTube. For example, Mech et al. (2015), have documented the hunting behaviours of wolves with respect to various prey types. Another facet of hunting techniques exists in modern warfare such as found in the Navy with surface and sub-surface (i.e., submarine) hunting methods. The development of evasion techniques has been studied as methods to avoid capture, and these are documented in the same manuals that study hunting techniques. For modern day land warfare, Wilson (2002) documents the evasion techniques used by the Special Air Service (SAS) to avoid capture while being pursued.

Current Research

Autonomous Agent Design Theory and Architectures. Research in Autonomous Agents has led to several design architectures, amongst them is GAIA, NetLogo, and Belief-Desire-Intent (BDI). While each has strengths and weaknesses, one of the major challenges of any AA architecture is in its implementation.

GAIA. Using NetLogo, Wilhelmy et al. (2014) structured the code for their MAS environment (hunter-prey scenario) according to the GAIA architecture. The result was lists of procedures roughly organised under individual agent types (hunter and prey) and functions (such as establishing the Contract Net). While the GAIA architecture worked for a small MAS environment (ie. hundreds of lines of code), it is questionable how scalable it is to larger code sets (ie. thousands of lines of code).

Grey Wolf Optimization (GWO). A hunting algorithm that predicts the convergence of omega wolves on a prey.

Hunting Agent (HA). This is an AA which hunts Prey Agents as a food source.

Lion Optimization Algorithm (LOA). A hunting algorithm that predicts the convergence of a beta lioness on a prey.

Multi-Agent Systems (MAS). This is a grouping of AAs which interact amongst each other, typically to accomplish a common goal.

Observer. This is the human that is establishing the World Agent parameters and running the MAS environment.

Prey Agent (PA). This is an AA which hunts stationary objects (such as apple trees) as a food source.

World Agent (WA). This is an AA which takes input from the Observer, establishes the MAS environment, and brokers the interactions between the various PAs and HAs throughout the runtime.

Summary

A review of the literature gave indications for the establishment of selection criteria for both the design architecture and environment in the evaluation of existing search, hunt, and evasion algorithms. The selection of an appropriate AA architectural model and MAS design environments is important because the HAPEE is going to need to be highly scalable. Also important is the ability to reuse code within the design environment in the instantiation of different types of agents. The ability to reuse code is also key to the scalability of the HAPEE model otherwise the code size could become repetitive, large, and unmanageable. The existing research also identified that a design environment should be chosen that can handle any type of

search, hunting, and evasion algorithms. With regards to selecting algorithms for pack leadership, the current research indicated that this could be addressed if we instantiate basic hunting procedures for the pack leaders.

the HAs. Thus, only the visual, auditory, and touch senses were used in evaluating the hunting algorithms.

Sight. The sense of sight was enabled with two visual scanning algorithms which were developed for the agents. The first was a reactionary algorithm and the second was an analytical algorithm. They were both originally developed to enable the prey to find their plant food source (ex: apple tree), they were then adapted to the hunters. Both algorithms, described in PA_perceive_scan1 and PA_perceive_scan2, use the NetLogo commands of patch-ahead, patch-left-and-ahead, and patch-right-and-ahead to sequentially scan the ground in front (and left and right) of the Agent until a visual limit is reached. As shown in Figure 4 below, the patch-left-and-ahead establishes the left arc of the visual field and the patch-right-and-ahead establishes the right arc. The scan starts in the middle (from near to far distance) and sweeps left (from middle) and then right (from middle) by incrementing first by 5 degrees and then 10 degrees (for the remainder). The algorithms stop scanning along a heading if it encounters an object (such as a rock feature or tree feature) that the agent should not be able to see through. The advantage of the algorithm is that it is simple to implement, the disadvantage is that it covers the squares (closest to the agent) several times while missing squares at distance (thus creating blind spots).

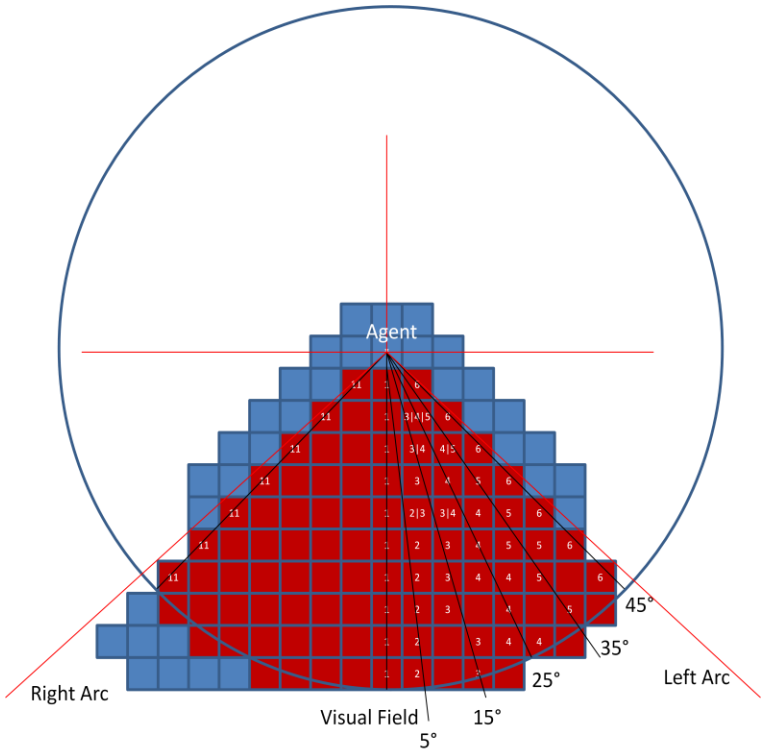

Figure 4: Visual Scanning Algorithm

The analytical scanning algorithm saves the objects perceived within the visual field of the agent and stores these in a perception array. The data in the perception array is then analysed later through any Belief Review Function (in this case PA_brf_food within the PA_brf_cycle) which uses it to identify patches of a certain colour.

Sound. The sense of hearing was enabled in the HAs through the HA_action_talk-prey_found procedure which would start the cycle of sound propagation with the WA procedure WA_brf_sounds-prey_found (which stores the sound in a State array that the WA initializes during setup through WA_brf_sounds-current-initialize and WA_brf_sounds-previous-initialize). The WA updates the sounds every Deliberation Cycle through the WA_brf_sounds-update_Current-to-Previous procedure and brokers these sounds with the Patch

Agents through the WA_Patch_brf_sounds procedure. This last procedure determines the residual sound level at the patch on which a PA or HA are occupying and lets them know whether a sound can be heard. The sound attenuation is given in Figure 5 below where a Level 1 sound (representing a whisper) can be heard on an adjecent square and a Level 7 sound (very loud) can be heard 200 square away. Note that this sound propagation algorithm can be adjusted to conform to any type of physical environment (as deserts and jungles have very different sound propagation properties).

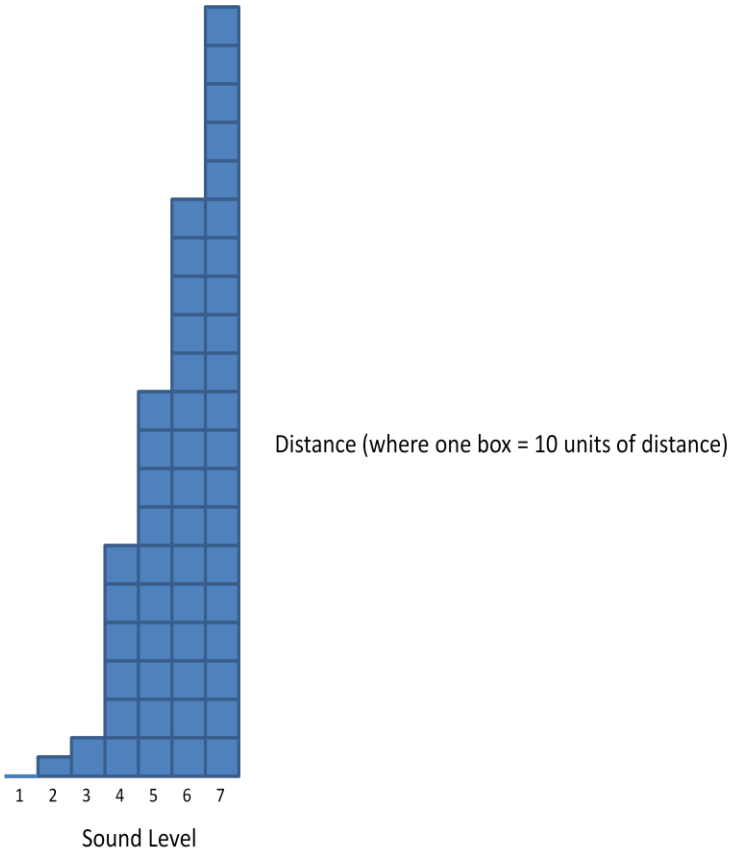

Figure 5: Graph of Sound Level vs Distance.

Smell. Each PA (through the GA) is given a unique scent (with GA_State_Scent-Type) and a scent level (with GA_State_Scent-Level). As a PA moves around the

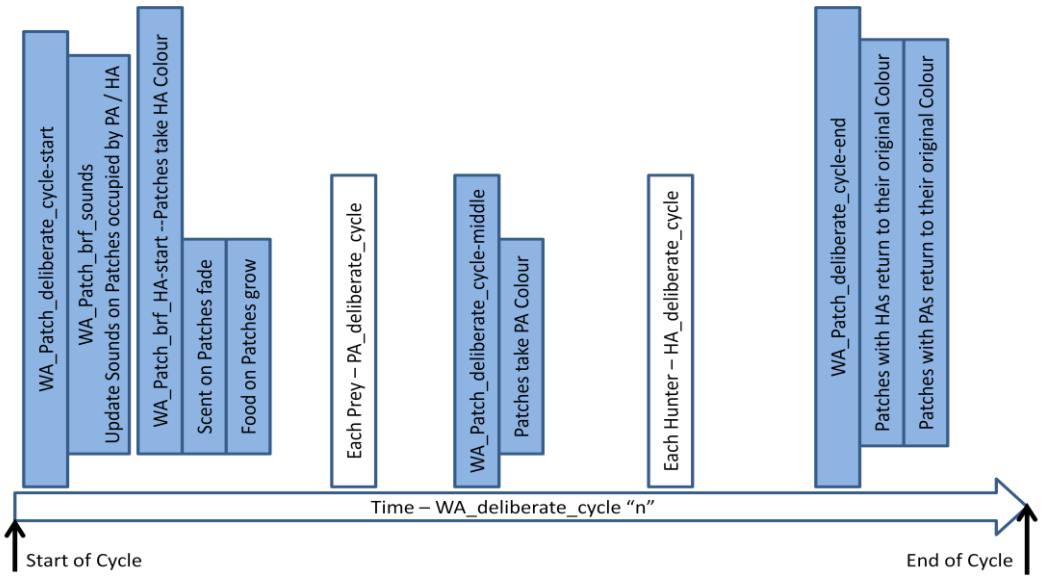

Figure 9: World Agent Deliberation Cycle.

All the HAs then step through their Deliberation Cycles followed by a return of all patches to their original colours. This cyclical process is detailed in Figure 9.

Generic Agent. The GA was developed to reuse code that is shared across all AAs regardless if they are PAs or HAs. While structured as a BDI AA, it is never instantiated in HAPEE. The GA is used to establish Design states as well as Belief states related to normal animal functioning (hunger, thirst, energy levels, etc.)

Prey Agent. The PA was designed around a simple Rest-Search-Eat cycle (Figure 10) which is activated by the Desire for food or water (GA_State_Energy-Current and GA_State_Water-Current).

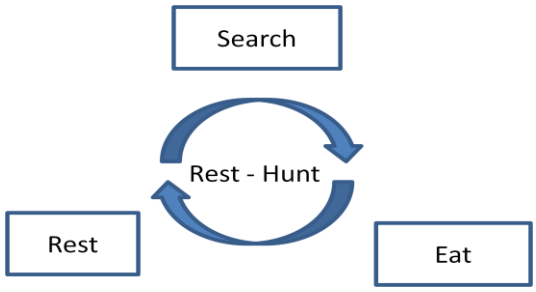

Figure 10: PA Activity Cycle.

While the PA has knowledge of the direction of the river, it searches for food based on a circular pattern of movement (like that of a partridge). When a PA encounters an HA, its evasion technique is simply to turn 180 degrees away from the HA and run away until exhausted.

Hunter Agent. The HA has a more complicated activity cycle as seen in Figure 11. Starting from Rest, the HA becomes hungry and begins the search for food. If it hears another wolf has found prey, then it will converge on that wolf. If it sees prey then it will alert the wolves as to its presence. Once the wolves converge on the Alpha wolf, they will encircle and then approach the prey. Once they get close enough, they will strike the prey and kill it. Eating terminates the cycle and the wolves return to a state of rest.

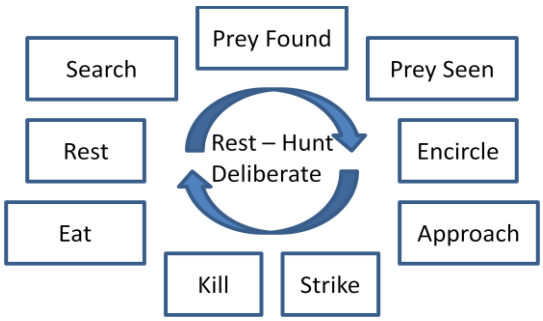

Figure 11: HA Activity Cycle.

Applied to one wolf, this cycle may seem simple. However, it is complicated by the fact that it controls the cohesion and the sequencing of the pack. This is particularly true for the encircle, approach, and strike phases of the hunt. To synchronize the activities of the wolves (or the lionesses), the code was developed around a hunting state (HA_Belief_Hunt-Mode) which is used to control all the HAs involved in the hunt. This detailed control process is depicted in Figure 12. While HAs can be at different states, the transition from one state to the other is dependent on the disposition of one HA with respect to the other (particularly the Alpha).

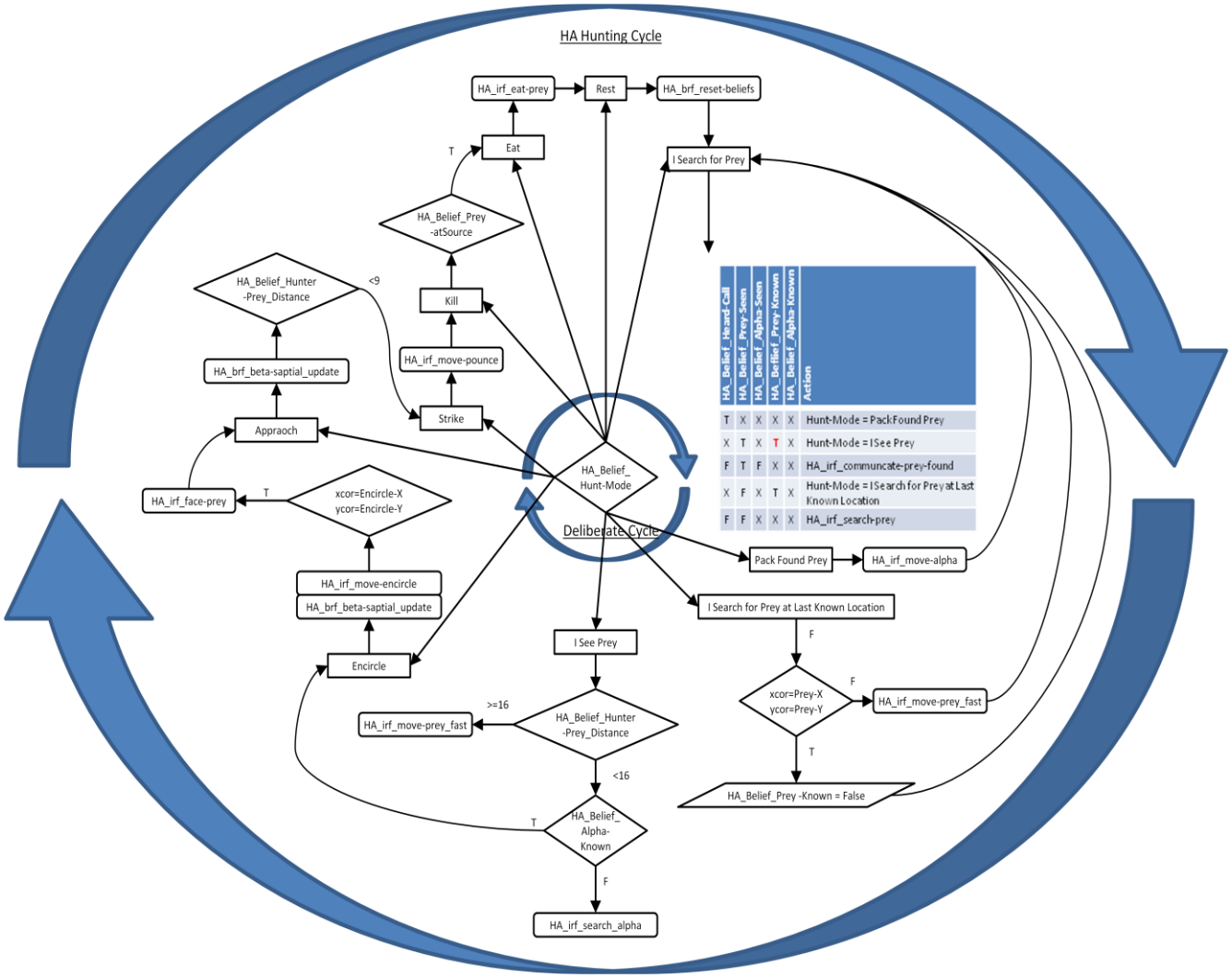

Figure 12: HA State-Based Hunting Cycle.

For instance, the wolves (or Beta lioness) will not transition to the strike phase unless the Alpha is seen striking the prey first. As discussed later, the Alpha typically strikes the prey from behind, thus initiating the strike from the other wolves (or Beta lioness). So, the Alpha roughly controls the pack, not through the use of messages, but through a state based process which controls the phasing of the overall hunting cycle.

ambiguity through the GWO. Thus, the GWO (equation 3.7) sums each of the position vectors from the Omega to the Alpha and Betas and then divides this by a factor of 3.

CHAPTER IV

RESULTS

BDI Architecture for NetLogo

With over four thousand lines of code for this project, the architecture provides a structured framework which is highly scalable. Organized into five major sections, additional sections could easily be added. As per Appendix-B, the code started with the declaration of variables for each of the five agent types: globals (for the World Agent), patches-own (for the Patch Agent), turtles-own (for the Generic Agent), hunted-own (for the Prey Agent), and hunters-own (for the Hunter Agent). The second section was the World Agent, the third section the Generic Agent, the fourth section the Prey Agent, and the final section was the Hunter Agent.

Each agent was sub-structured along its BDI functions where these were ordered alphabetically to simplify look-up. The sub-sections for each agent were structured as follows:

a. Actions;

b. Belief Review Function (BRF);

c. Deliberation;

d. Desire Review Function (DRF);

e. Intention Review Function (IRF);

f. Perceptions; and

g. Plans. Note that this section was only used for the World Agent. The GA, PA, and HA called Actions directly from the IRF.

The BDI Architecture proved conducive to supporting code reuse through the Generic Agent.

Perceptions and Beliefs. Three methods were found of handling the way perceptions updated beliefs. The first method had two distinct functions, the Perceptions and the Belief Review Function, with a clear delineation between the functions. In this case changes to variables occurred within their respective functions but were copied to the variables in the other function to process them (ie. modify them). The second method combines the Perception variables directly into the BRF. The third method, opposite to the second method, has the Belief variables being directly updated within the Perception functions. The advantage of the first method is that code is separated and evolution of the perception functions can occur independently from the evolution of the belief review functions. Another advantage is that the results of a perception cycle can be used by several BRFs. The disadvantage is a requirement for additional memory and computational inefficiency due to the belief review functions having to examine essentially the same information an additional time to then change the beliefs; this increases the amount of code required and introduces additional latency (due to increased processing requirements). The second and third methods intertwine the variables of the Perception and Belief Review Functions, which has the advantage of being more computationally efficient and requiring less code. The disadvantage of these two methods is that the intertwined code becomes more difficult to evolve as they must be advanced together (there is no longer a clean interface between the two). Another disadvantage is that the results of a perception cycle cannot be used by more than one BRF; each BRF needs to run its own perception cycle. With a lot of BRFs, this would also become highly inefficient.

Lion Optimization Algorithm. Graph 1 plots both the results of the LOA and GWO and fits the linear curve to the LOA. The equation of the fitted LOA curve is calculated to be y = -13x + 117.

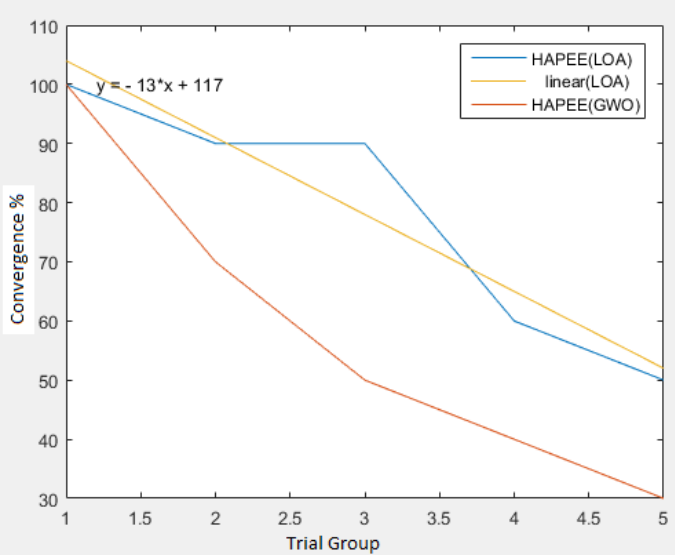

Figure 14: LOA fitted with a Linear Curve.

Grey Wolf Optimizer. Graph 2 plots both the results of the LOA and GWO and fits the linear curve to the GWO. The equation of the fitted LOA curve is calculated to be y = -17x + 109.

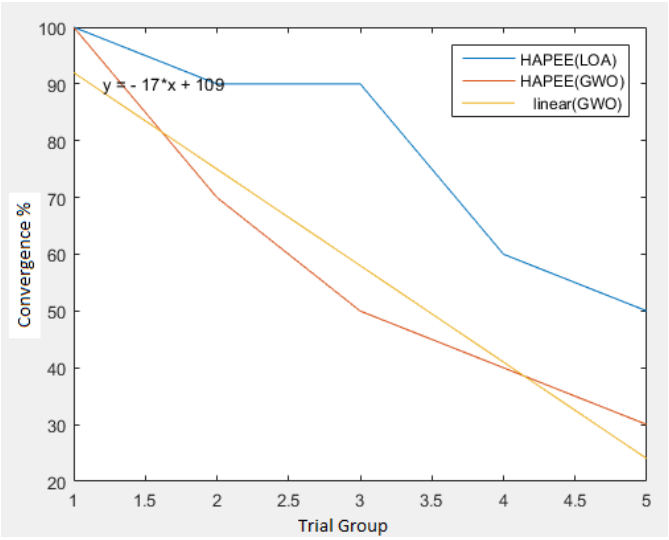

Figure 15: GWO fitted with a Linear Curve.

the near field as it passed over the same square several times. A more thorough visual scan could be developed that inversed the pattern to that of a radial scan that then incremented from near-to-far (ie. a radial scan would occur at a fixed distance and then restarted at an incremented distance).

CHAPTER VI

CONCLUSION AND RECOMMENDATION

Conclusion

This research found that a BDI architecture enabled the NetLogo environment to effectively generate a HAPEE where two algorithms (LOA and GWO) were evaluated with the LOA being found to be more resilient to obstacles than the GWO.

BDI Architecture

As hypothesized, the Modified BDI Architecture could provide significant structuring to the NetLogo code thus making it feasible for a large-scale MAS environment.

NetLogo MAS Design Environment

Also, hypothesized, with the structuring provided by the Modified BDI Architecture, the NetLogo environment has proven to be able to establish a large-scale MAS Hunting Algorithm Performance Evaluation Environment (HAPEE) which is flexible and scalable.

Algorithms

The HAPEE found that the Lion Optimization Algorithm (LOA), in the presence of obstacles, was more effective at hunter convergence than the Grey Wolf Optimization algorithm. It also found that the Grey Wolf Optimization algorithm, after having lost sight of the prey, was more capable of re-acquiring and converging on the prey than was the Lion Optimization Algorithm. The LOA either succeeded or failed, whereas the GWO had a larger variance where it could recover from failure. Thus, as hypothesized, the HAPEE can be used to provide an alternative means to test Hunting Algorithms.

Suggestions for Further Research

Several areas of further research were alluded throughout this thesis paper. While the BDI Architecture is addressed in step with the NetLogo MAS Design Environment, the Algorithms are dealt with separately.

BDI Architecture and NetLogo MAS Design Environment

Learning vs. Dexterity. The simple code structuring of Perceptions vice Beliefs gives indications as to the natural selection of higher intelligence vice faster reflexes. As an analogy, the analytical algorithm favours a learning agent (with a scaling number of BRFs) while a reactionary algorithm favours an agent with high dexterity (aka agility). This simple design choice perhaps gives some indication as to the construct of the human mind. Humans with higher analytical and learning abilities often have lower dexterity than those humans that are very good at sports (high dexterity), but face challenges with their analytical thinking and abilities to learn diverse subjects. For further study, animals, that are not constantly under the threat of being hunted, are naturally selected to evolve their minds from those of high reflex ability to those that have a higher learning ability. Also, a third visual scanning algorithm could exist which would be a hybrid of the analytical and reactionary algorithms, this could be developed as a follow-on activity.

Temporal Equivalence. The BDI Architecture does not overtly address the temporal equivalence of a Deliberation Cycle amongst AAs (or within a MAS). There are good reasons for this, mainly because the BDI Architecture is meant for the design of AAs within the real World (not a World Agent) where time is real

Wolf Hunting Results

Trial Number	Vegetation Density (%)	Tick-Count until Convergence	Convergence	dT
0	0	19	Y	n/a
1	1	-	N	-
2	1	19	Y	0
3	1	-	N	-
4	1	29	Y	+10
5	1	19	Y	0
6	1	19	Y	0
7	1	19	Y	0
8	1	-	N	-
9	1	19	Y	0
10	1	19	Y	0
1	2	-	N	-
2	2	16	Y	-3
3	2	-	N	-
4	2	20	Y	+1
5	2	-	N	-
6	2	20	Y	+1
7	2	19	Y	0
8	2	-	N	-
9	2	-	N	-
10	2	20	N	+1
1	3	18	Y	-1
2	3	-	N	-
3	3	-	N	-
4	3	-	N	-
5	3	-	N	-
6	3	33	Y	+14
7	3	-	N	-
8	3	62	Y	+43
9	3	19	Y	0
10	3	-	N	-
1	4	-	N	-
2	4	-	N	-
3	4	75	Y	+56
4	4	-	N	-
5	4	29	Y	+10
6	4	-	N	-
7	4	32	Y	+13
8	4	-	N	-
9	4	-	N	-
10	4	-	N	-