Evolved Communication Strategies and Emerged Behaviour of Multi-Agents in Pursuit Domain

Gina Grossi Brian J. Ross

Brock University

Department of Computer Science

St. Catharines, Ontario,

Canada L2S 3A1

gina.grossi4@gmail.com bross@brocku.ca

Outline

- Problem overview
- Background
- System
- Experiment details
- Results
- Conclusions

More details:

Gina Grossi, Learning Strategies for Evolved Co-operating Multi-Agent Teams in the Pursuit Domain
Master's thesis, Dept of Computer Science, Brock U., 2017.

Problem

- Evolving co-operating multi-agent teams is a difficult problem researched extensively over the years.
- An appropriate test bed for multiagent systems is the predator-prey pursuit problem (the pursuit domain).
- This research adds to previous work investigating how genetic programs (GPs) can be used in a predator – prey scenario to allow agents to learn to communicate.

Research in Multi-Agent Systems

- Comprehensive survey

Pannait & Luke, 2005

- Coordinating agents

Haynes *et al.* 1995; Denzinger & Fuchs, 1996

- Communicating agents

Yanco & Stein, 1993; Iba, 1998; Kam-Cheun & Giles, 2000; Reverte *et al*. 2008

- Emergent behaviours using GP

Zhang & Cho 2000; Tanev *et al*. 2005

- Learning strategies in game environments

Luke *et al.* 1997; Alhejali and Lucas, 2011; Cardona *et al*. 2013; Kou *et al*. 2013

... and many more.

Motivation

- Investigate how well genetic programs can influence learning using different communication protocols.
- Task agents with learning the meaning of commands.
- Define how well evolved predator agents can use a generic command language to learn the behaviour of tracking prey.

Applications

- Behaviour strategies of multi-agents is a central issue in multi-agent systems research.
- Can be applied to many real world applications in which agent co-ordination is necessary (e.g. Robots working together to complete a task).
- Possible foundation for using learning algorithms in developing game AI behaviours.

Limitations

- The GP language is limited in order to allow high-level behaviours to emerge.
- Communication is synchronized (messages are sent and received in a sequential order).
- Strongly-Typed GP is used to control the top-level tree structure.

Communication Strategies

(Pannait and Luke, 2005)

Communication Strategies	Types	Communication Channel
Direct	Hard Coded: *use pre-defined commands* Learned Language: *learn meaning of commands*	Message Board or Message Passing
Indirect	Implicit transfer of information: *from agent to agent through modification of the environment. Inspired by insects social use of pheromones.*	Footstep trail Breadcrumb trail Hints through object placement

Communication Strategies

(Pannait and Luke, 2005)

Communication Strategies	Types	Communication Channel
Direct	Hard Coded: *use pre-defined commands* Learned Language: *learn meaning of commands*	Message Board or Message Passing
Indirect	Implicit transfer of information: *from agent to agent through modification of the environment. Inspired by insects social use of pheromones.*	Footstep trail Breadcrumb trail Hints through object placement

Learning Strategies

(Pannait and Luke, 2005)

Learning Strategies	Description	Number of Agent Learners	Individual Fitness	Common Fitness
Team	Homogeneous	1	No	Yes
	Heterogeneous	1 or more	No	Yes
Concurrent	Fully co-operative: *agents always work together*	1 or more	No	Yes
	Partially co-operative: *agents sometimes work together*	1 or more	Yes	Yes
	Competitive: *agents compete with each other (Co-evolution)*	1 or more	Yes	Yes

Learning Strategies

(Pannait and Luke, 2005)

Learning Strategies	Description	Number of Agent Learners	Individual Fitness	Common Fitness
Team	Homogeneous	1	No	Yes
	Heterogeneous	1 or more	No	Yes
Concurrent	Fully co-operative: *agents always work together*	1 or more	No	Yes
	Partially co-operative: *agents sometimes work together*	1 or more	Yes	Yes
	Competitive: *agents compete with each other (Co-evolution)*	1 or more	Yes	Yes

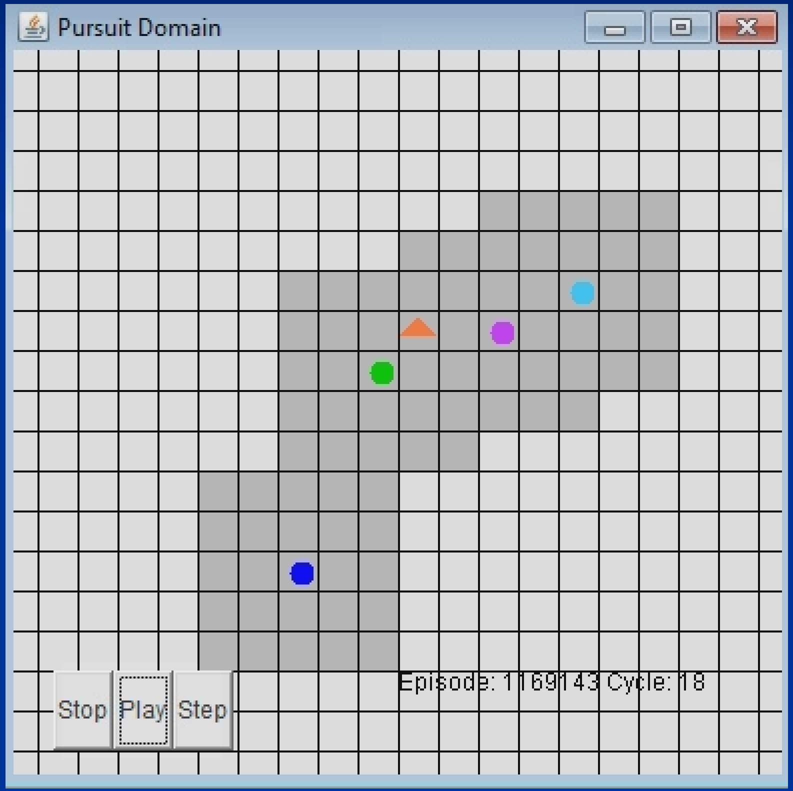
Approach

- Implement a fully co-operative, heterogeneous team based learning strategy with a global fitness measure.

- Adopt a communication strategy using a learned language consisting of generic commands *C0 & C1.*

Pursuit Domain Package (PDP)

Kok & Vlassis, 2003



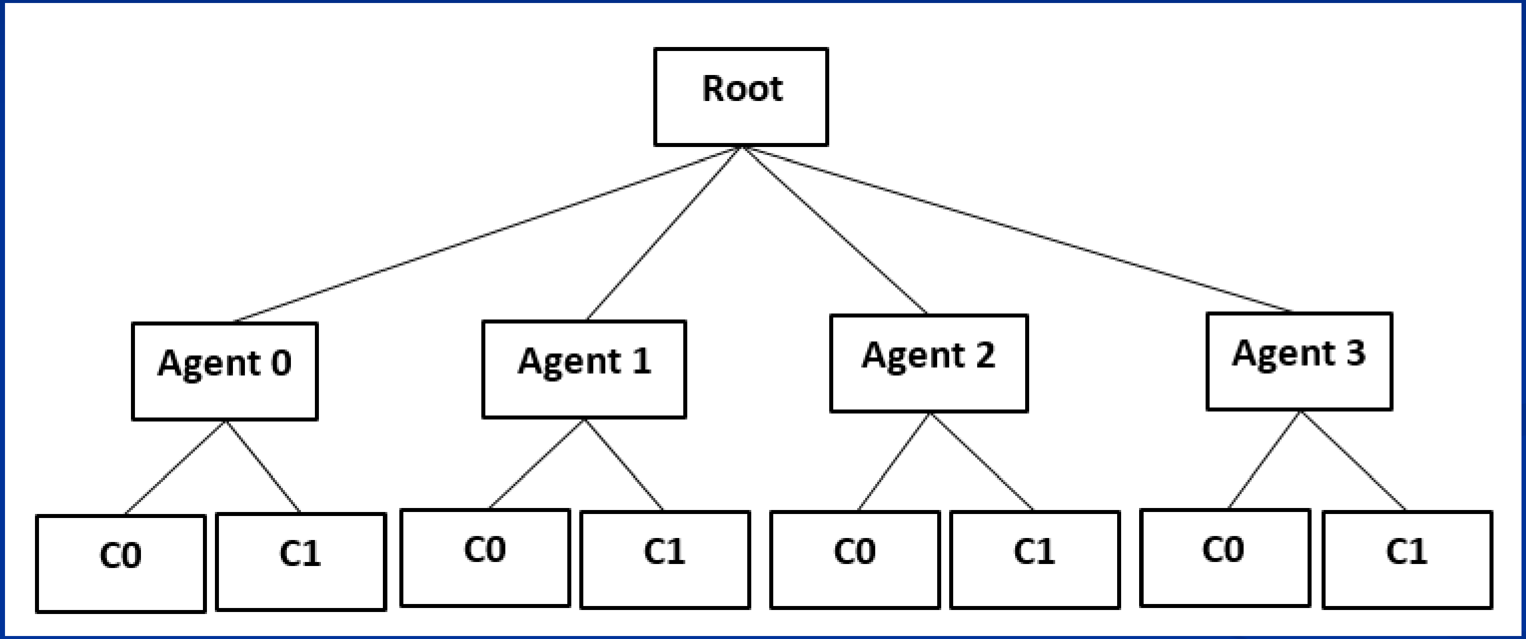
1 Prey Agent (Orange Triangle).
4 Predator Agents (each represented by a circle)

- Infinite 20 x 20 grid containing 4 predators and 1 prey.
- Predator agents: *work together to find and track prey.*
- Prey agent: *evades the predators.*
- Agents move one step (Up, Down, Left and Right) per time cycle. *(Note: agents movements can wrap around edges)*
- Field of View (FOV) = 2

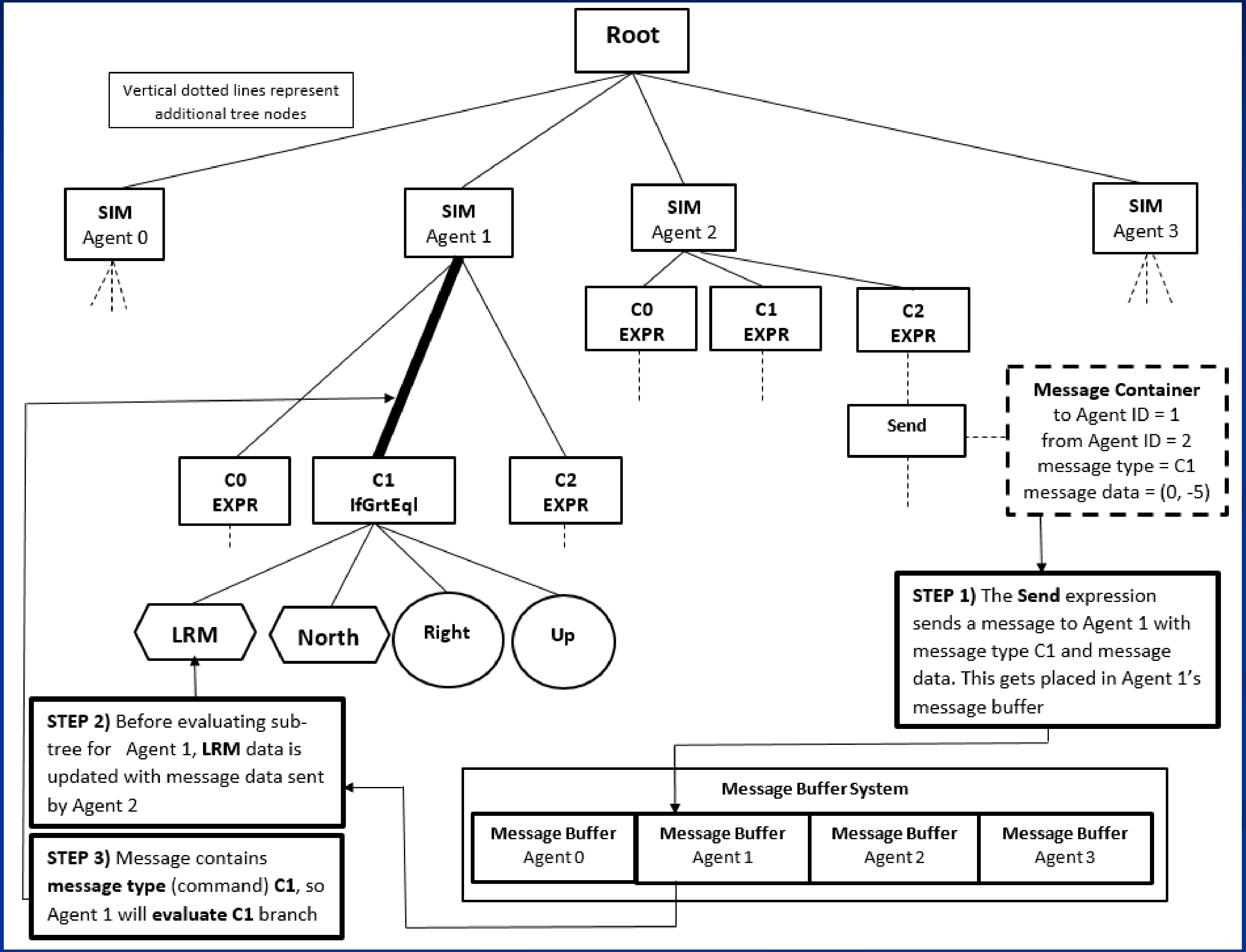
GP

- Java Evolutionary Computation (ECJ) System :
  - https://cs.gmu.edu/~eclab/projects/ecj/
  - strong typing is used so that each predator agent evolves its own sub-tree

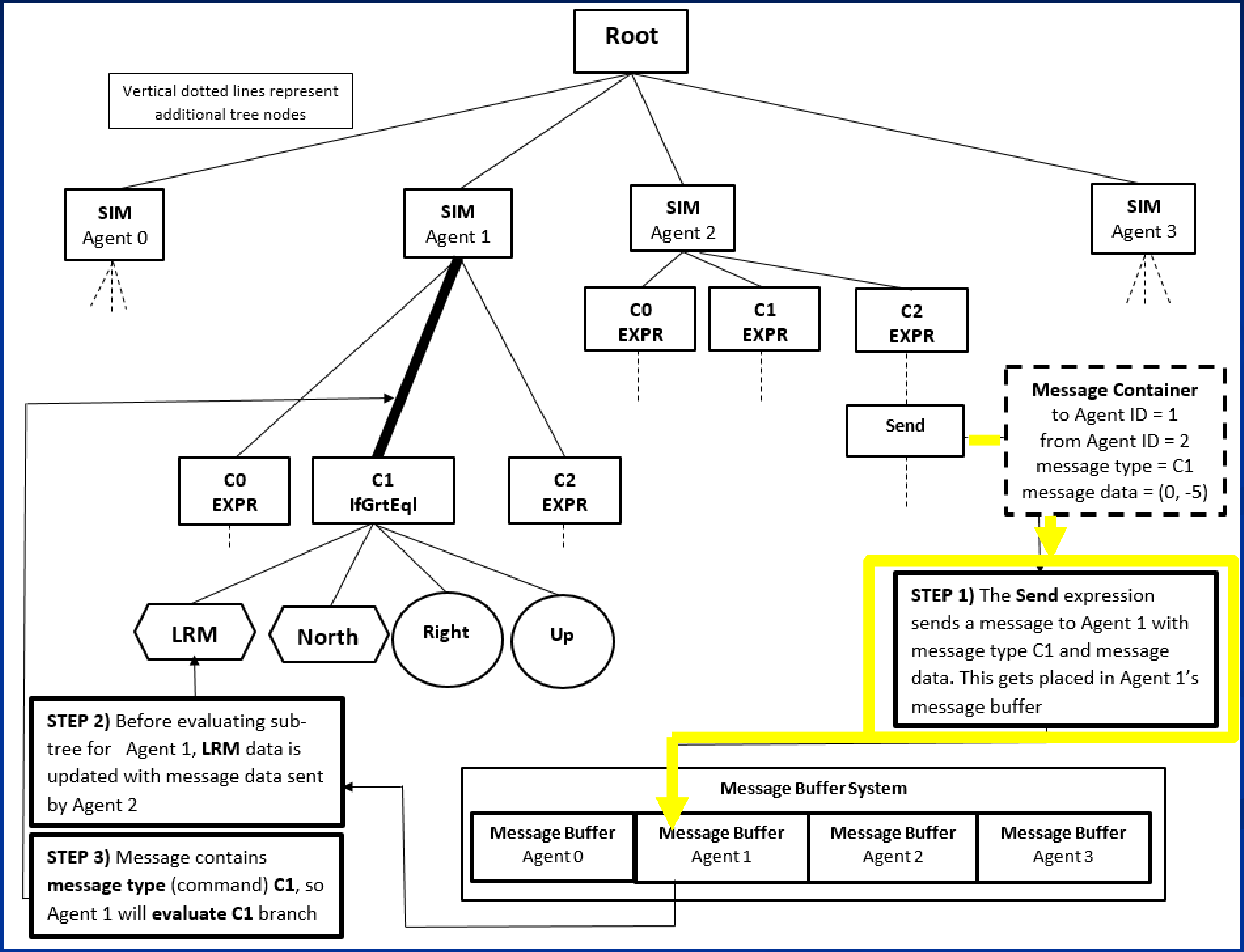
GP Top-level Tree for Predator Agents



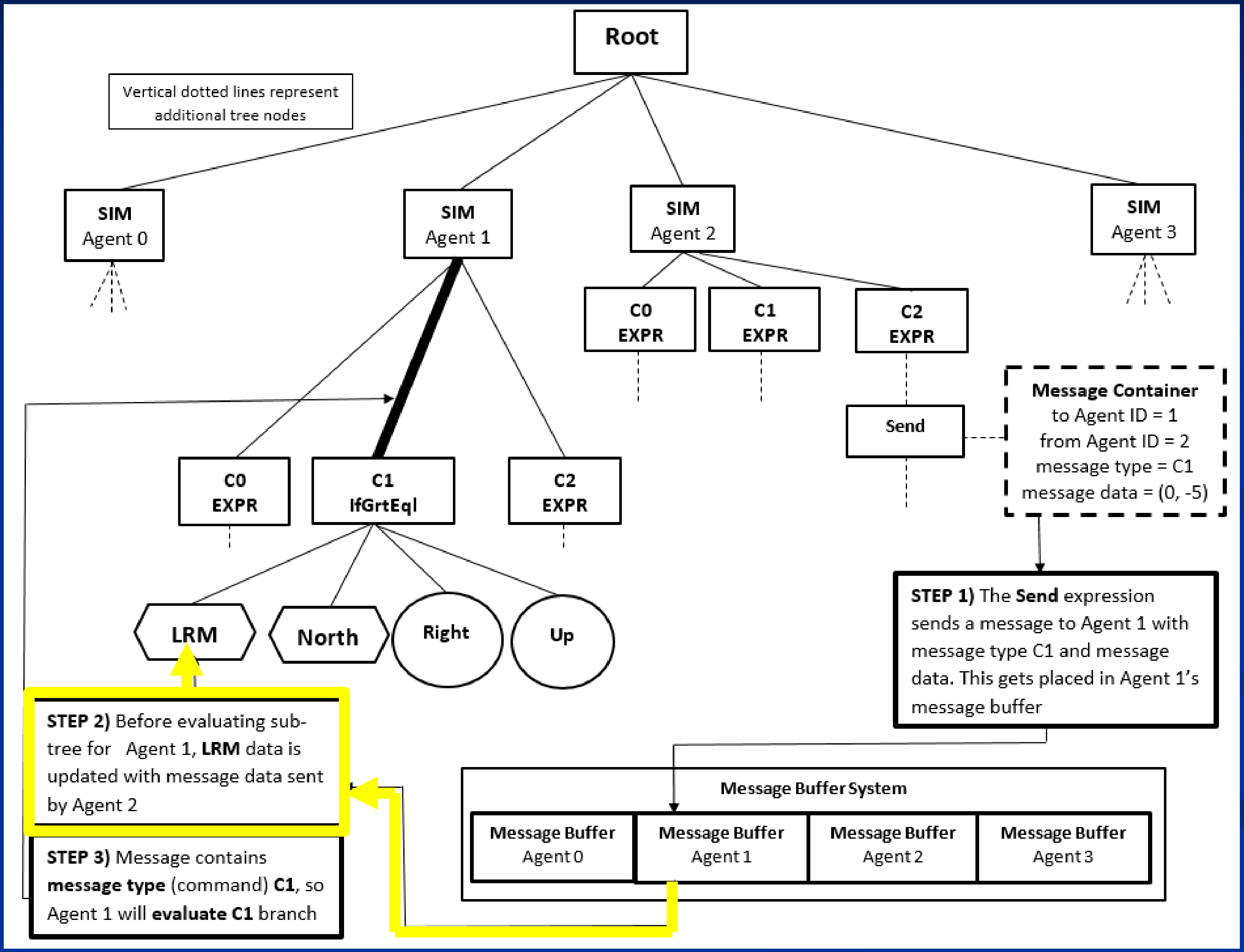
System Overview



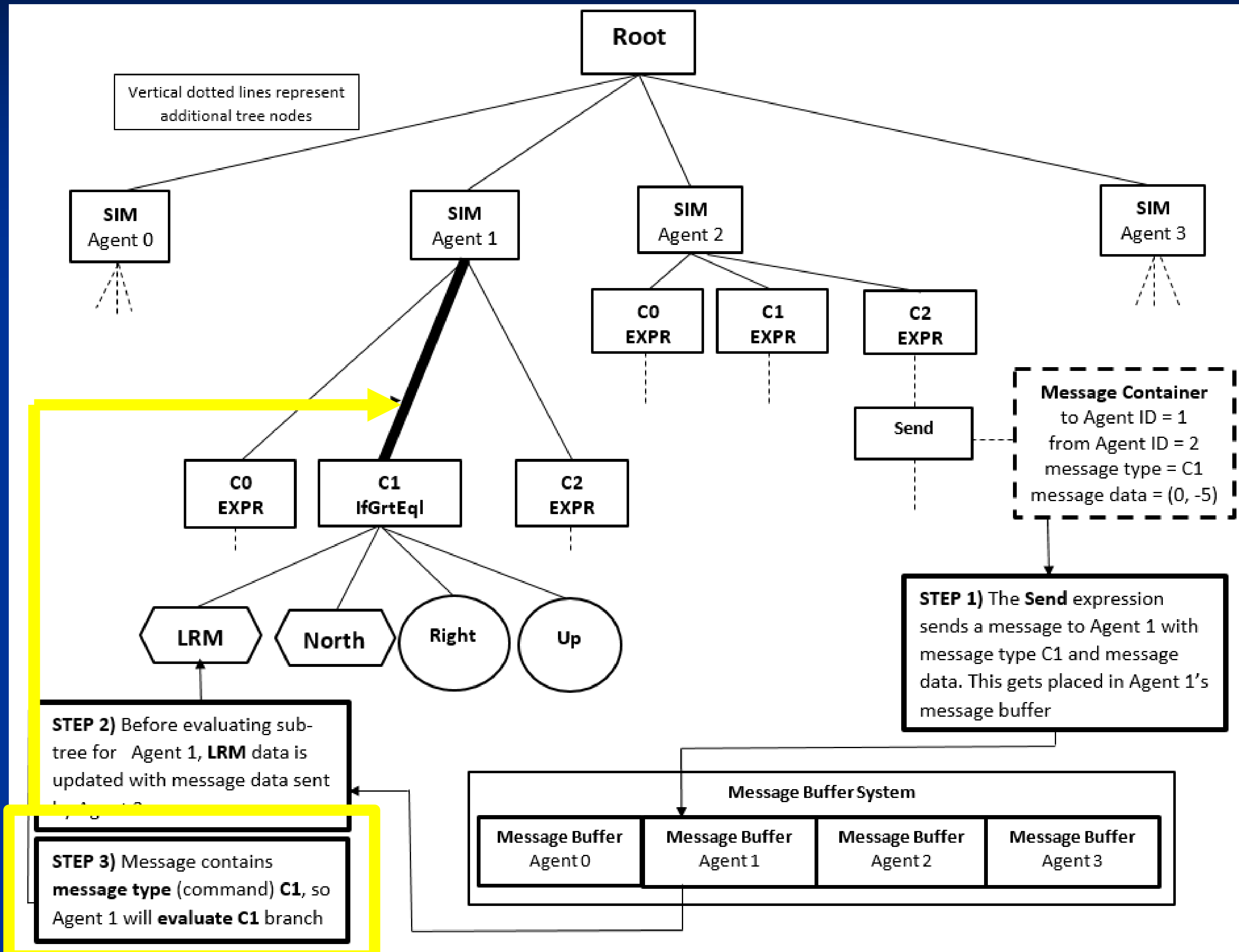
System Overview



System Overview



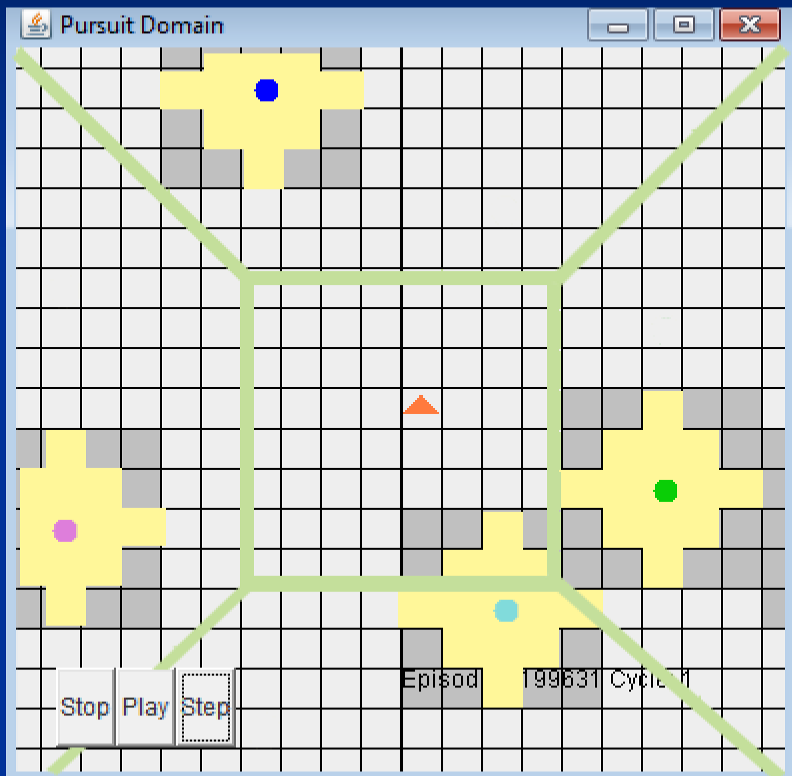
System Overview



Communication Protocols

Communication Protocols	Method of Message Passing
Send22	A0 $\leftrightarrow$ A1 A2 $\leftrightarrow$ A3
Send21	A0 $\rightarrow$ A1 A2 $\rightarrow$ A3
SendLine	A0 $\rightarrow$ A1 $\rightarrow$ A2 $\rightarrow$ A3
SendLine2D	A0$\leftrightarrow$A1$\leftrightarrow$A2$\leftrightarrow$ A3
Send13	A0 $\rightarrow$ A1, A2, A3
SendAll	A0 $\rightarrow$ A1, A2, A3 A1 $\rightarrow$ A0, A2, A3 A2 $\rightarrow$ A0, A1, A3 A3 $\rightarrow$ A0, A1, A2
SendK (similar to Iba [7])	
SendKN0	A $\rightarrow$ nearest agent
SendKN1	A $\rightarrow$ 2nd nearest agent
SendKN2	A $\rightarrow$ farthest agent

Components of Fitness Measure



Orange Triangle = Prey Agent in its starting area
Coloured Circles = Predator Agents, each in their starting areas

- Cycle = 1 time unit and 1 movement step on the grid per agent.
- Episode = 30 cycles. The total time for which agents are allowed to track prey.
- Starting Positions: Each agent starts in their own designated start area
- Details of Training : Each training run consists of 10 episodes with each agent starting at a random position within their area.
- Details of Testing: Each test run consists of 30 episodes with each agent starting at the same position.

Fitness Measure

- Fitness is measured by finding the sum of episode fitness scores.
- The episode fitness is the sum of each of the agent's distance to the prey in 30 cycles where each cell is 1 unit of distance.
- GP individuals with better fitness scores will minimize the distance sum as agents track the prey.

Fitness Measure

$$TotDist = \sum_{k=1}^{q} \sum_{j=1}^{m} \sum_{i=0}^{3} \sqrt{(A_i.x - P.x)^2 + (A_i.y - P.y)^2}$$

Training fitness, *TotDist*. A_i represents the location *of Agent*$_i$, where $i = 0 \ldots 3$, *P* is the location of the prey, *m* represents the number of cycles and *q* is the number of episodes. We set *q* to 10 in training.

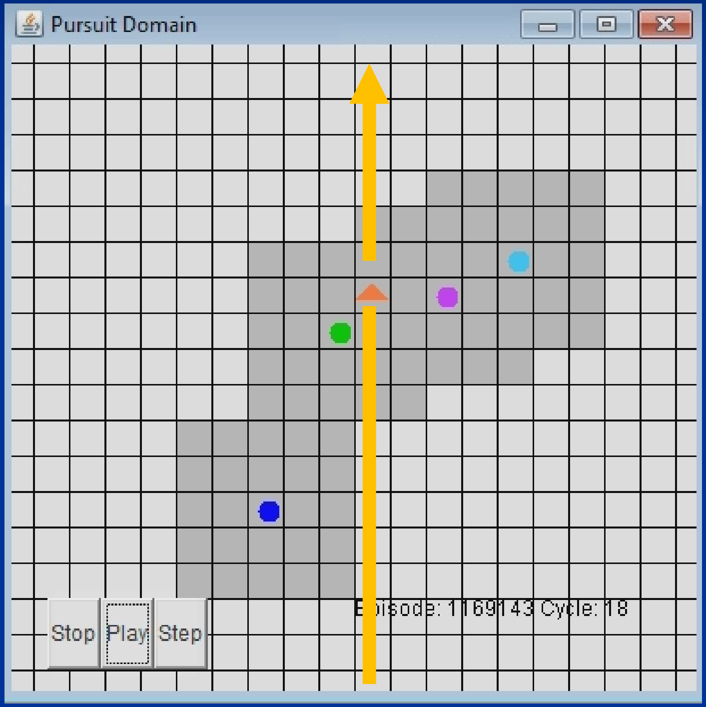
$$AveDist = \frac{TotDist}{q}$$

Testing fitness, *AveDist*, uses the *TotDist* to measure the average total distance of all test runs. We set *q* to 30 in testing.

Typical GP Parameters

GP Parameter	Value
Initial Tree Method	Koza's *Ramped half-and-half*
Min-Max Tree size (ramp)	4-6
Population size	1000
Generations	125
Selection	Tournament, size = 4
Crossover	90%
Mutation	10%
Runs per experiment	20

Prey Linear Movement



- Prey starts at a random position (within its start area).
- Moves in the Up direction, once each cycle (time step).

Prey Linear Movement Results

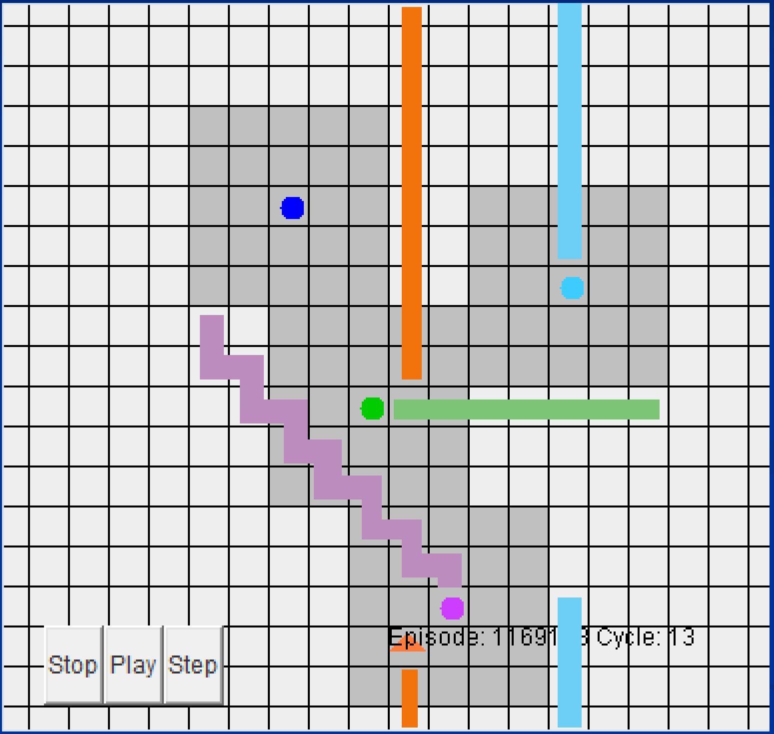
Prey Movement	Communication Type	Min Fitness of 20 runs	Ave Fitness of 20 runs	Max Fitness of 20 runs
Linear	SendAll	707	788	852
	Send22	727	789	878
	Send21	713	801	876
	Send13	712	803	849
	SLine2D	747	806	875
	SLine	728	804	920
	SendK	783	827	868

SendAll and Send22 are the top performers when the Prey moves linearly. The results for Send22 and SendAll are statistically different (95% confidence interval) than the worst performer SendK.

Evolved Behaviours

- Most test evolved competent agents that were able to follow the prey.
- Many of the communication protocols did not produce significantly different results.
- However, some experiments did regularly evolve interesting behaviours that show high-levels of co-ordination among agents.
- An emergent behaviour found in SendAll and Send22 protocols shows a synchronization of message sending that results in a staircase movement pattern to find the prey.

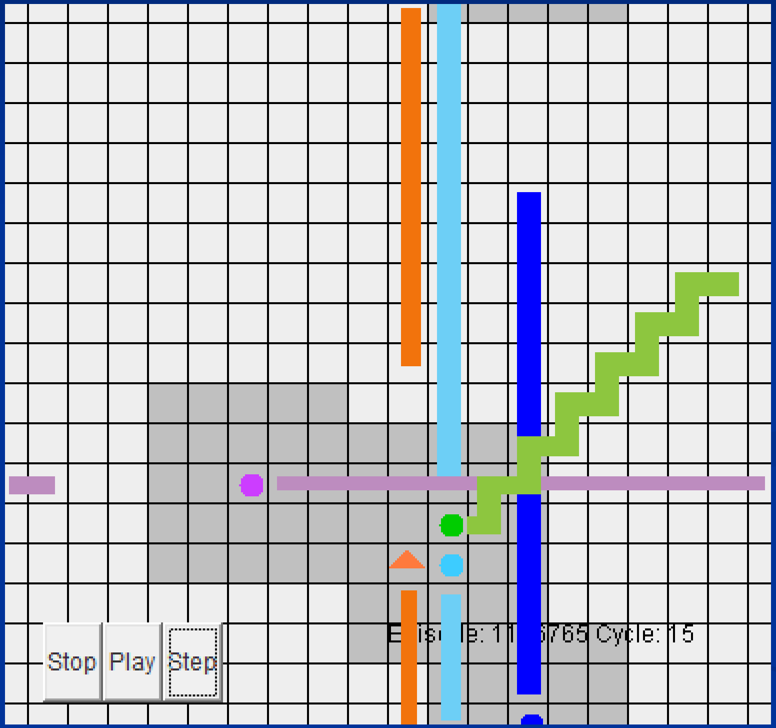
Emergent Behaviour: Synchronized Message Sending - *Staircase Pattern*



SendAll (Run 14)

Agent 1 (purple) moves in a staircase pattern until it finds the prey. Once it is in FOV it then tracks the prey.

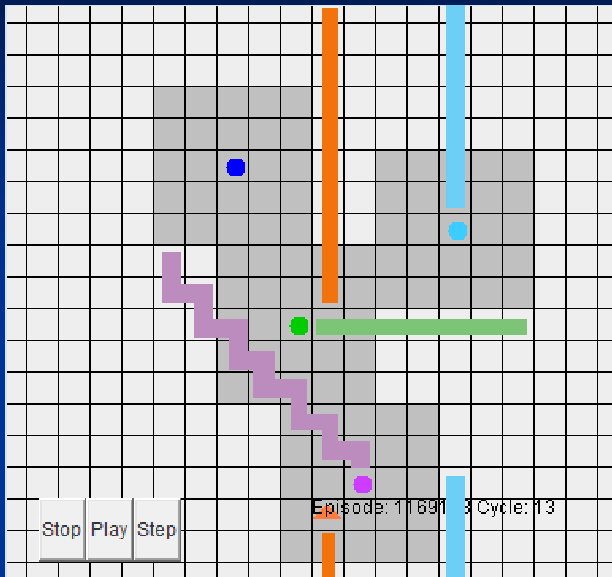
Emergent Behaviour: Synchronized Message Sending - *Staircase Pattern*

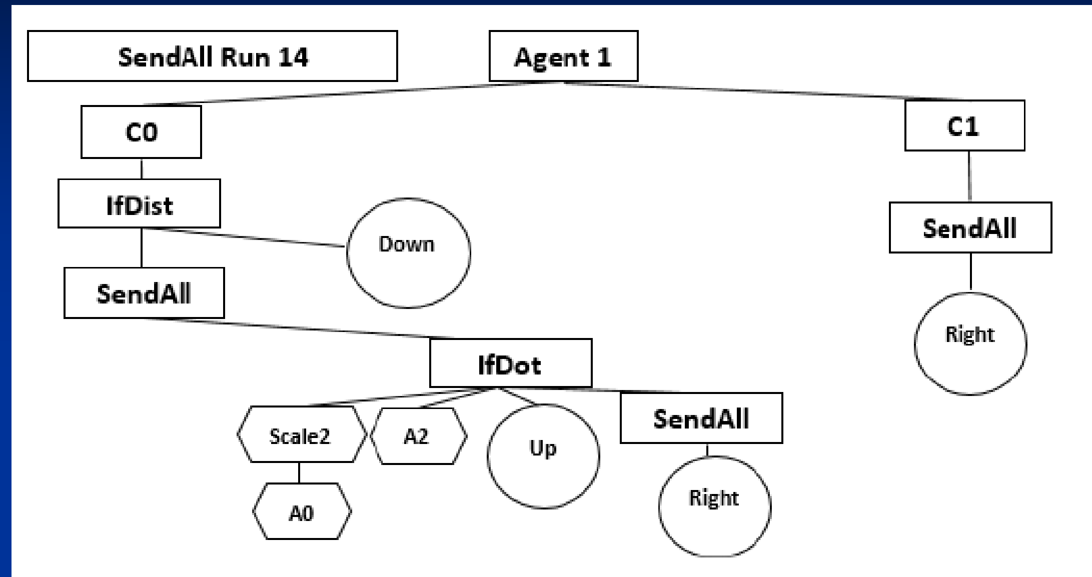


Send22 (Run 14)

Agent 3 (green) moves in a staircase pattern until it finds the prey. Once it is in FOV it tracks the prey.

Staircase Pattern SendAll (Run 14)



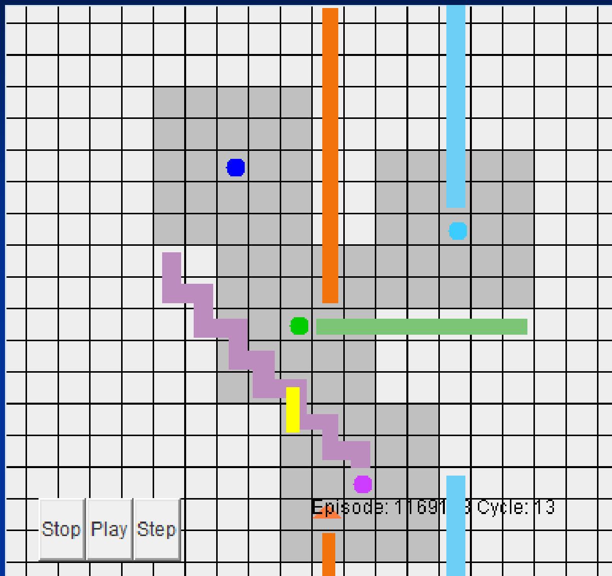


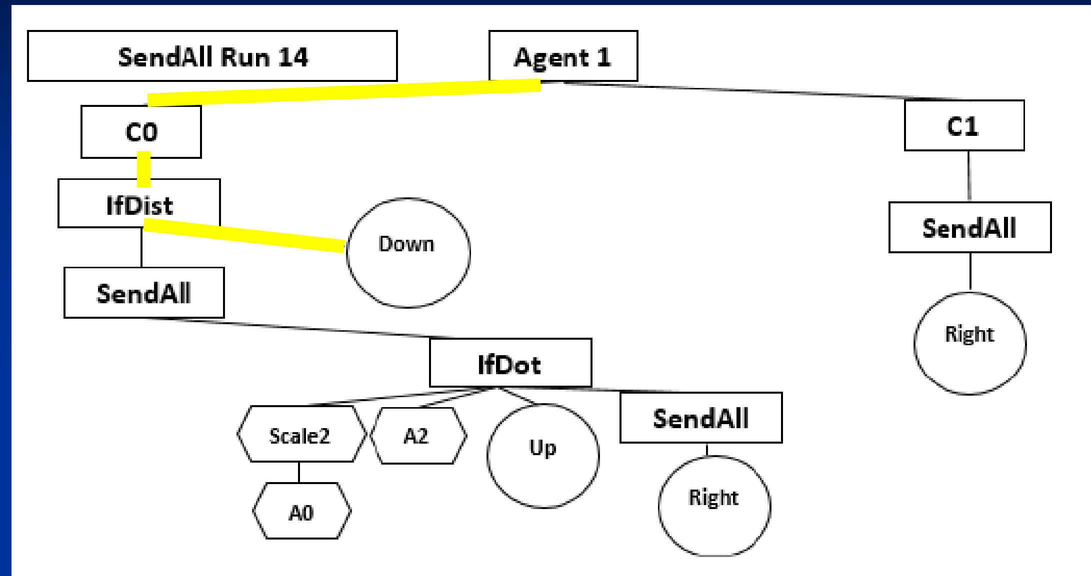
- Cycles 9 – 13: Agent 1 (purple) & Agent 3 (green) send messages to all other agents every other cycle.
- As a result, Agent 1 & 3 evaluate their C0 & C1 every other cycle.

SendAll(Linear) Test Run 14, Cycles 9-13

Cycle	From Agent	Agent 0 Message LRM	From Agent	Agent 1 Message LRM	From Agent	Agent 2 Message LRM	From Agent	Agent 3 Message LRM
9	1	(40,40)			1	(40,40)		
10	3	(40,40)	3	(40,40)	3	(40,40)	1	(40,40)
					1	(40,40)		
11	1	(40,40)			1	(40,40)		
12	3	(40,40)	3	(40,40)	3	(40,40)	1	(1,-6)
					1	(3,-11)		
13	1	(-4,-9)			1	(3,-11)	1	(1,-5)
					1	(3,-10)		

Staircase Pattern SendAll (Run 14)



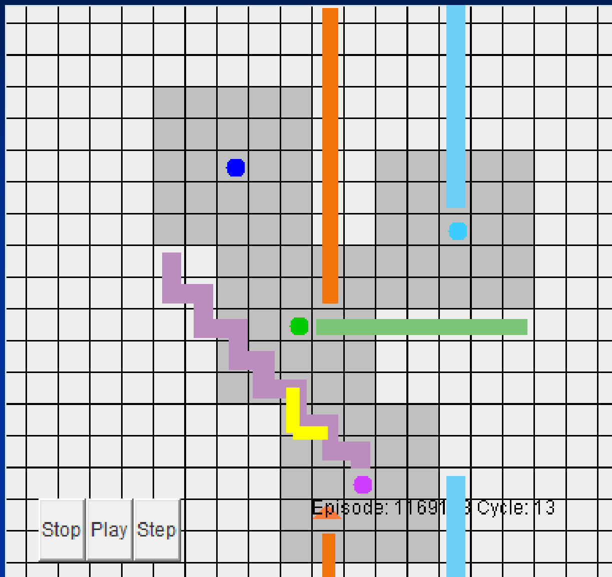


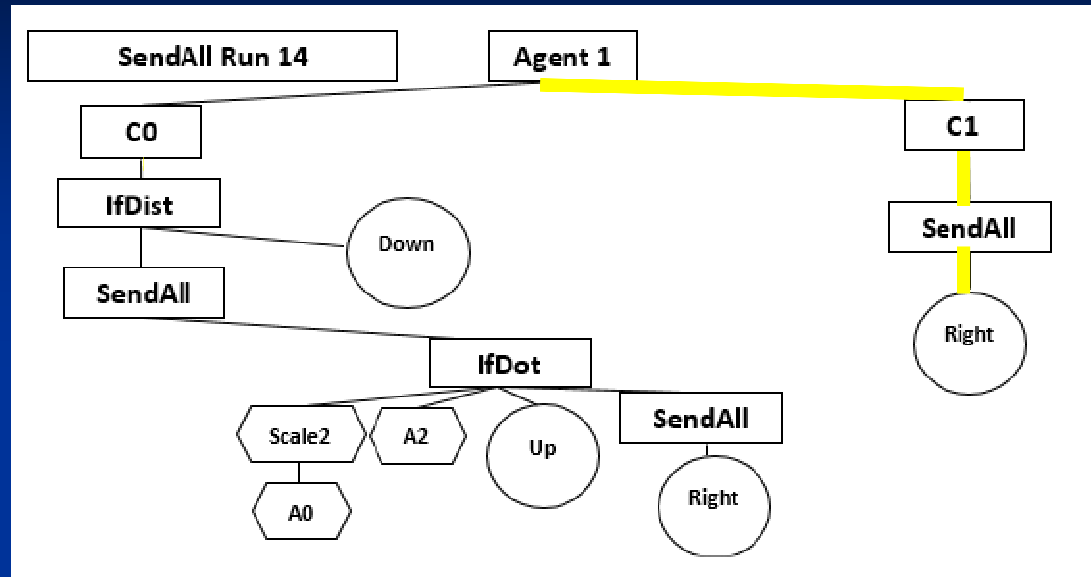
SendAll(Linear) Test Run 14, Cycles 9-13

Cycle	From Agent	Agent 0 Message LRM	From Agent	Agent 1 Message LRM	From Agent	Agent 2 Message LRM	From Agent	Agent 3 Message LRM
9	1	(40,40)			1	(40,40)		
10	3	(40,40)	3	(40,40)	3 1	(40,40) (40,40)	1	(40,40)
11	1	(40,40)			1	(40,40)		
12	3	(40,40)	3	(40,40)	3 1	(40,40) (3,-11)	1	(1,-6)
13	1	(-4,-9)			1 1	(3,-11) (3,-10)	1	(1,-5)

- Cycle 9 : Agent 1 has no message, evaluates C0, moves down.

Staircase Pattern SendAll (Run 14)



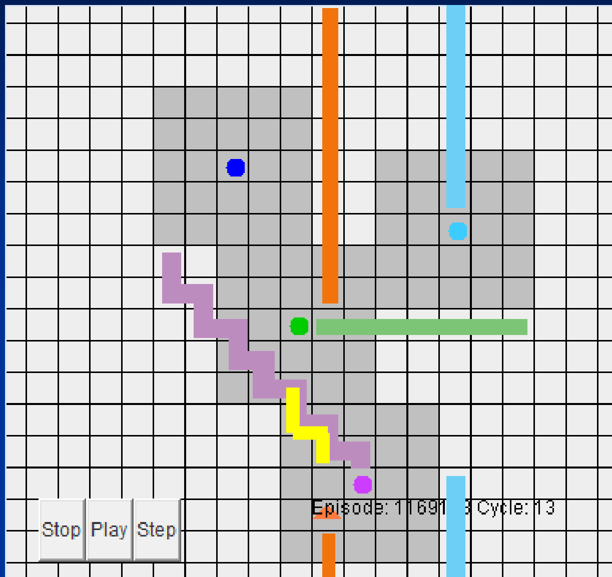


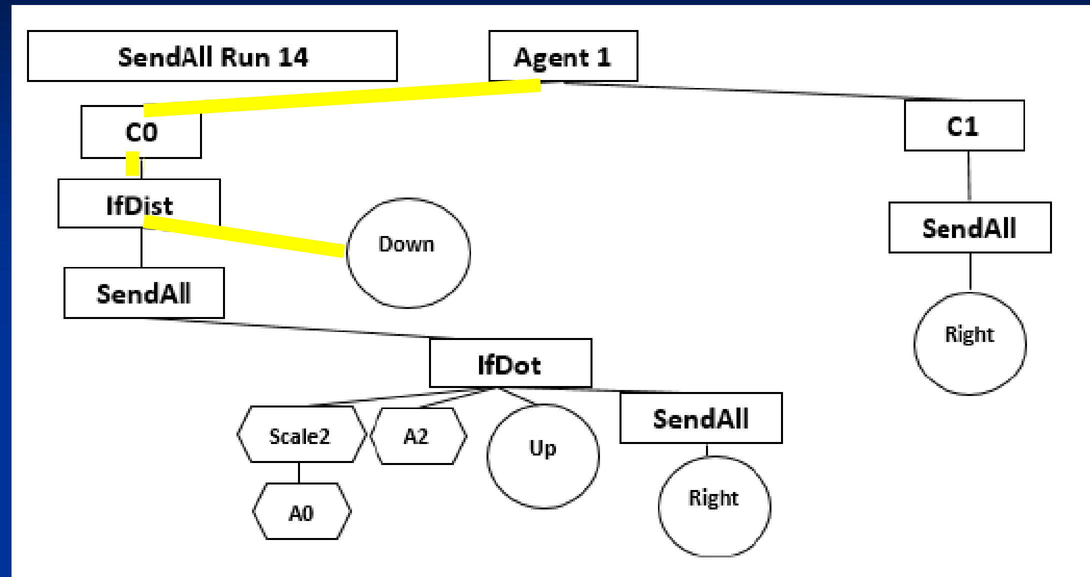
- Cycle 10 : Agent 1 has a message, evaluates C1, moves right.

SendAll(Linear) Test Run 14, Cycles 9-13

Cycle	From Agent	Agent 0 Message LRM	From Agent	Agent 1 Message LRM	From Agent	Agent 2 Message LRM	From Agent	Agent 3 Message LRM
9	1	(40,40)			1	(40,40)		
10	3	(40,40)	3	(40,40)	3 1	(40,40) (40,40)	1	(40,40)
11	1	(40,40)			1	(40,40)		
12	3	(40,40)	3	(40,40)	3 1	(40,40) (3,-11)	1	(1,-6)
13	1	(-4,-9)			1 1	(3,-11) (3,-10)	1	(1,-5)

Staircase Pattern SendAll (Run 14)



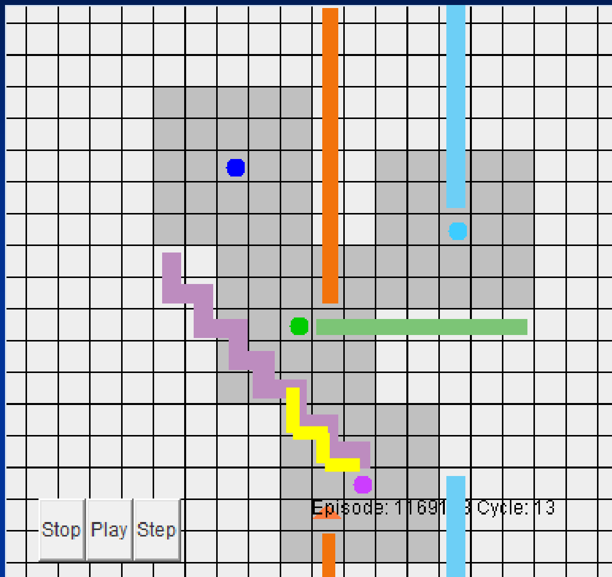


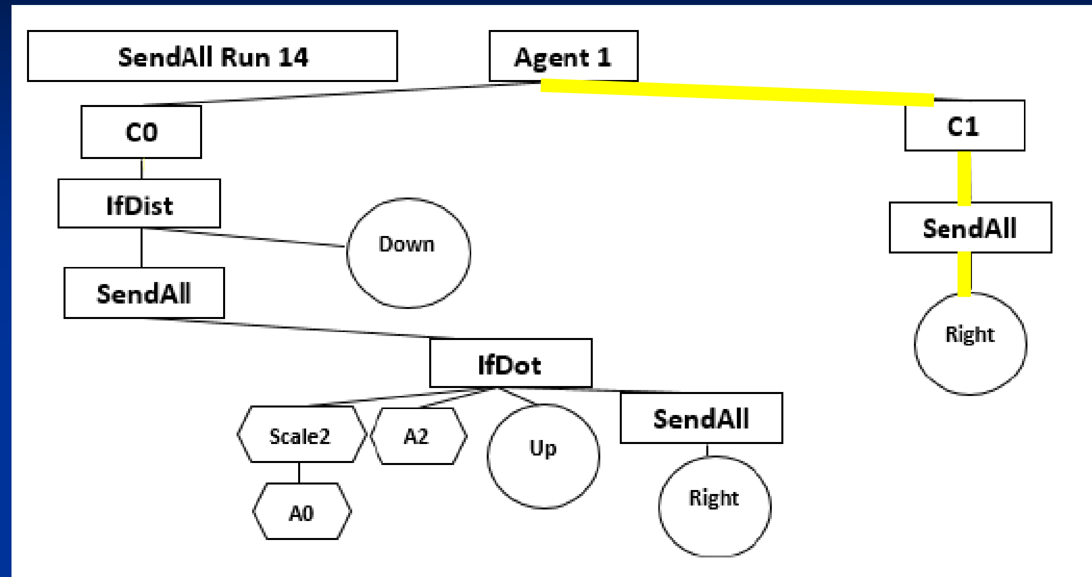
- Cycle 11: Agent 1 has no message, evaluates C0, moves down.

SendAll(Linear) Test Run 14, Cycles 9-13

Cycle	From Agent	Agent 0 Message LRM	From Agent	Agent 1 Message LRM	From Agent	Agent 2 Message LRM	From Agent	Agent 3 Message LRM
9	1	(40,40)			1	(40,40)		
10	3	(40,40)	3	(40,40)	3	(40,40)	1	(40,40)
					1	(40,40)		
11	1	(40,40)			1	(40,40)		
12	3	(40,40)	3	(40,40)	3	(40,40)	1	(1,-6)
					1	(3,-11)		
13	1	(-4,-9)			1	(3,-11)	1	(1,-5)
					1	(3,-10)		

Staircase Pattern SendAll (Run 14)



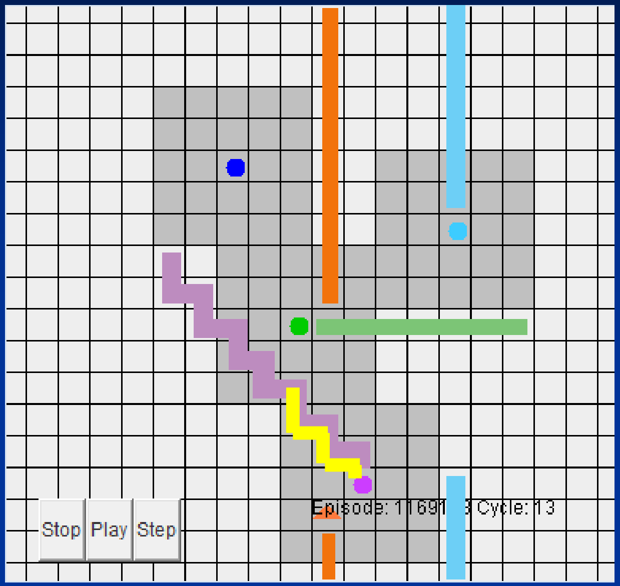


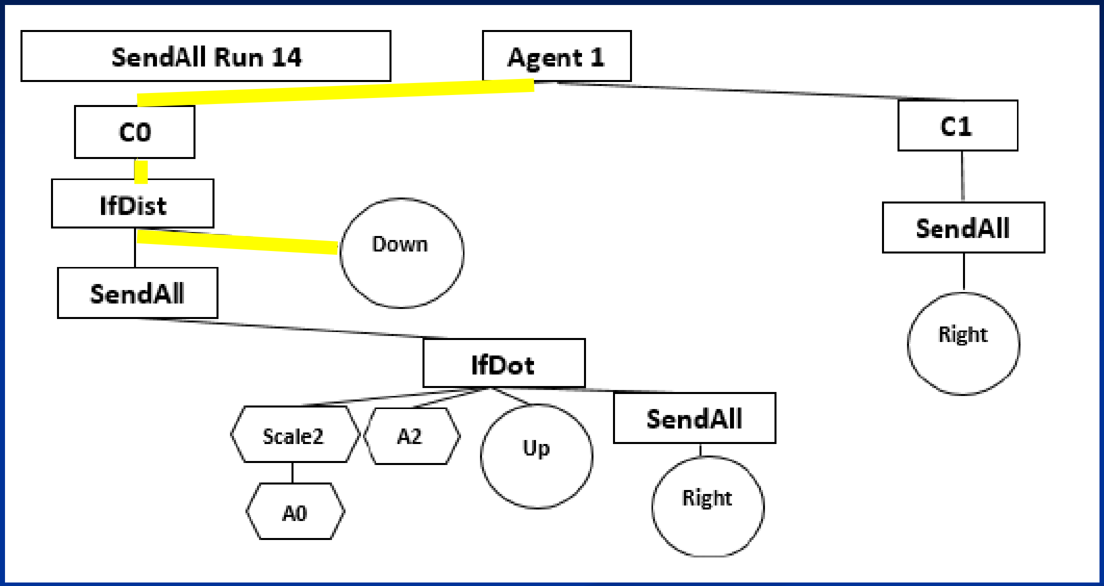
- Cycle 12: Agent 1 has a message, evaluates C1, moves right.

SendAll(Linear) Test Run 14, Cycles 9-13

Cycle	From Agent	Agent 0 Message LRM	From Agent	Agent 1 Message LRM	From Agent	Agent 2 Message LRM	From Agent	Agent 3 Message LRM
9	1	(40,40)			1	(40,40)		
10	3	(40,40)	3	(40,40)	3	(40,40)	1	(40,40)
					1	(40,40)		
11	1	(40,40)			1	(40,40)		
12	3	(40,40)	3	(40,40)	3	(40,40)	1	(1,-6)
					1	(3,-11)		
13	1	(-4,-9)			1	(3,-11)	1	(1,-5)
					1	(3,-10)		

Staircase Pattern SendAll (Run 14)





- Cycle 13: Agent 1 has no message, evaluates C0, moves down.

SendAll(Linear) Test Run 14, Cycles 9-13

Cycle	From Agent	Agent 0 Message LRM	From Agent	Agent 1 Message LRM	From Agent	Agent 2 Message LRM	From Agent	Agent 3 Message LRM
9	1	(40,40)			1	(40,40)		
10	3	(40,40)	3	(40,40)	3 1	(40,40) (40,40)	1	(40,40)
11	1	(40,40)			1	(40,40)		
12	3	(40,40)	3	(40,40)	3 1	(40,40) (3,-11)	1	(1,-6)
13	1	(-4,-9)			1 1	(3,-11) (3,-10)	1	(1,-5)

Video: Synchronized Message Sending - *Staircase Pattern*

Guard Reinforcement Behaviour

- A behaviour found in the video game series Metal Gear Solid (MGS) by Konami, is a guard protecting an area.
- Generally, a guard remains in one area to protect it. If he spots an intruder, reinforcements are called for backup.
- The guard and reinforcements track (and attack) the intruder.
- The best test run for the SendAll protocol evolved a simple form of this guard behaviour.

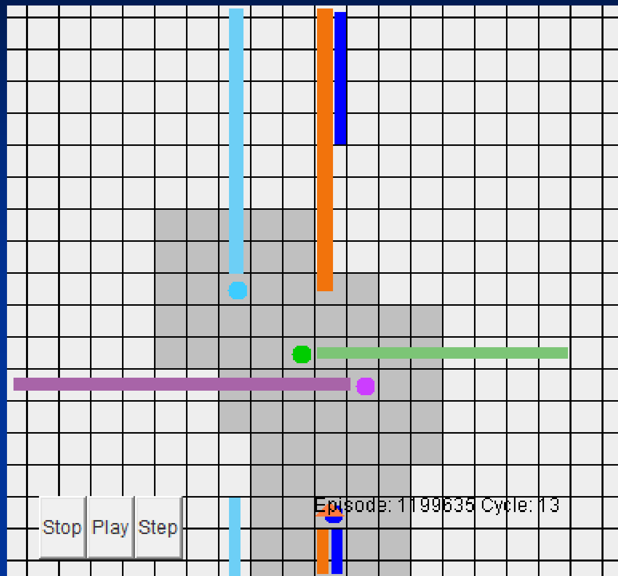
Video: Guard Reinforcement Behaviour

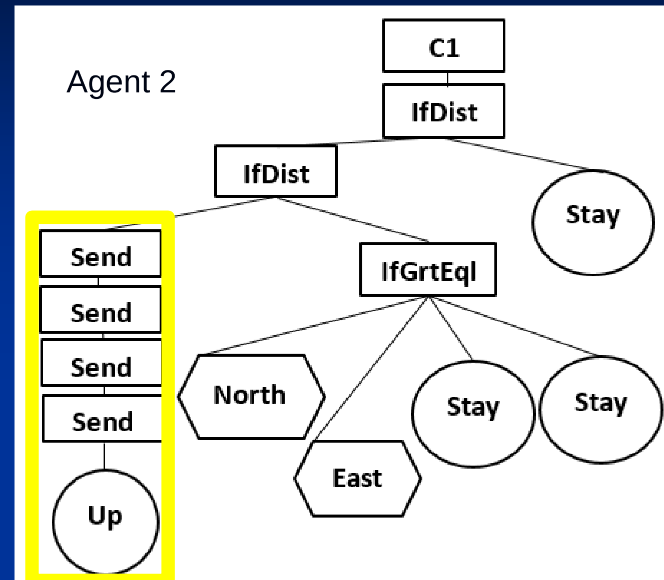
Metal Gear Solid (MGS)

https://www.youtube.com/watch?v=NnLBZFgFhZY

Guard Reinforcement Behaviour

SendAll, Run 15





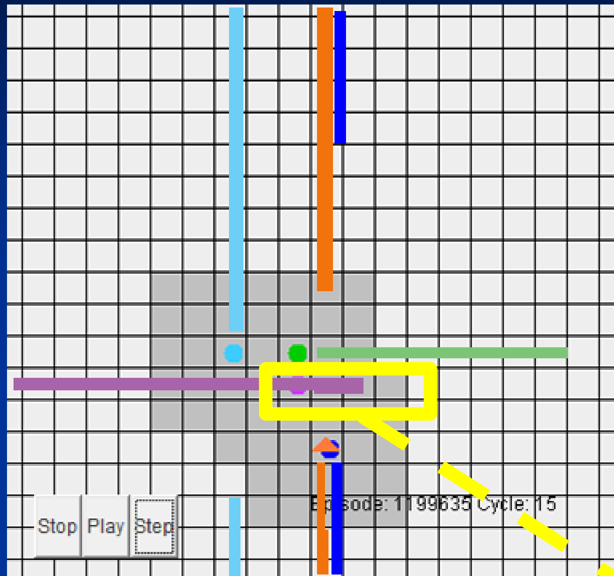
SendAll(Linear) Run 15, Cycles 4-17

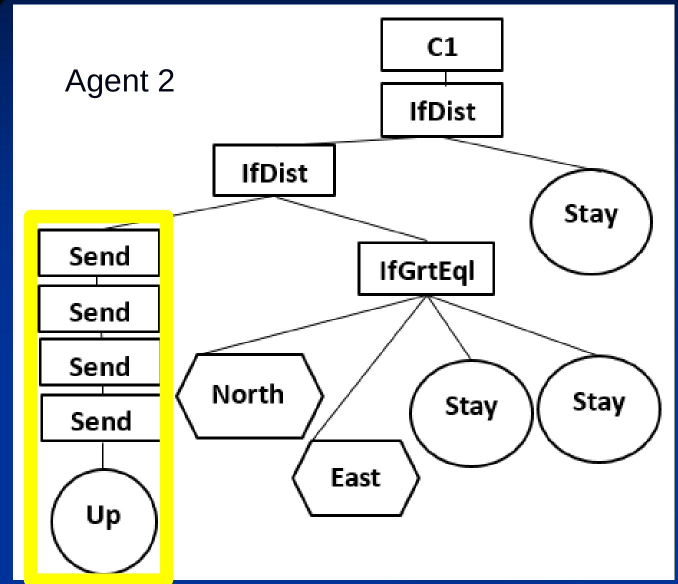
Cycle	From Agent	Agent 0 Message LRM	From Agent	Agent 1 Message LRM	From Agent	Agent 2 Message LRM	From Agent	Agent 3 Message LRM
4	3	(40,40)	0	(40,40)	0	(40,40)	2	(-7,6)
	3	(40,40)	3	(40,40)	3	(40,40)	2	(-7,6)
	3	(40,40)	3	(40,40)	3	(40,40)	2	(-7,6)
	3	(40,40)	0	(40,40)	0	(40,40)	2	(-7,6)
5	2	(3,-5)	2	(8,7)	0	(40,40)	2	(-6,7)
	2	(3,-5)	3	(40,40)	3	(40,40)	2	(-6,7)
	3	(40,40)	3	(40,40)	3	(40,40)	2	(-6,7)
	3	(40,40)	0	(40,40)	0	(40,40)	2	(-6,7)

- Agent 2 (dark blue) acts as a guard. Stays in position until in FOV of prey.
- Cycles 4 - 17: Sends out 4 messages when in FOV of prey and begins to follow prey.

Guard Reinforcement Behaviour

SendAll, Run 15





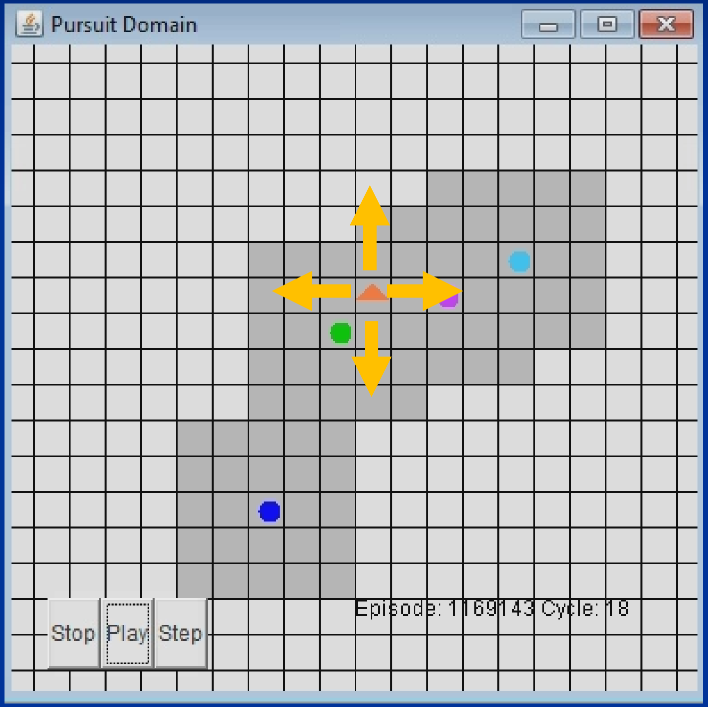
- Agent 2 (dark blue) sends out 4 messages to all agents.
- Cycles 14 - 15: Agent 1 (purple) changes direction and moves left, Agent 3 (green) waits.

SendAll(Linear) Run 15, Cycles 4-17

Cycle	From Agent	Agent 0 Message LRM	From Agent	Agent 1 Message LRM	From Agent	Agent 2 Message LRM	From Agent	Agent 3 Message LRM
15	2	(3,-5)	2	(0,-3)	0	(40,40)	2	(1,-3)
	2	(3,-5)	3	(40,40)	3	(40,40)	2	(1,-3)
	3	(40,40)	3	(40,40)	3	(40,40)	2	(1,-3)
	3	(40,40)	0	(40,40)	0	(40,40)	2	(1,-3)
17	2	(3,-1)	2	(2,-1)	0	(40,40)	2	(1,-1)
	2	(3,-1)	3	(40,40)	3	(40,40)	2	(1,-1)
	3	(40,40)	3	(40,40)	3	(40,40)	2	(1,-1)
	3	(40,40)	0	(40,40)	0	(40,40)	2	(1,-1)

Video: Evolved Guard Reinforcement Behaviour

Prey Random Movement



- Prey starts at a random position (within its start area).
- Moves randomly in one of the directions each cycle (time step):

Up ↑ Down ↓ Left ← Right →

Prey Random Movement Results

Prey Movement	Communication Type	Min Fitness of 20 runs	Ave Fitness of 20 runs	Max Fitness of 20 runs
Random	SendAll	638	720	776
	Send21	683	729	805
	SLine2D	682	730	763
	SLine	674	735	774
	SendK	672	735	772
	Send13	673	735	767
	Send22	698	739	803

SendAll is the top performer when the Prey moves randomly. However, all protocols perform equally as well as each other because the results are not statistically different (95% confidence interval).

Emergent Behaviour: Synchronized Message Sending

- The message buffers for the top performers for SendAll and for Send22 showed that there is also an emergence of a synchronized message pattern similar to previous experiments.
- However, evolved agents could not account for all four movement directions of the prey.
- The unpredictable movement of the prey caused agents to easily move out of view of the prey.

Conclusions

- Some experiments did regularly evolve interesting behaviours that show high-levels of co-ordination among agents.
- Emergent behaviour of synchronized message sending using generic commands help agents find prey.
- Guard and reinforcement behaviour in best result resembled scripted guard behaviour in game.
- Synchronized message sending not effective with random moving prey.

Conclusions

- Future work
  - Expand the GP language and fitness for random moving prey.
  - Test solution in different variants of the pursuit domain
    - Different grid sizes (30x30, 40x40 … 100x100)
    - Increase # of cycles per episode
    - Allow diagonal movement
  - Test solution in alternate predator-prey scenario (e.g. Ms. Pac-Man)