PSO for Competitive Co-evolution of Neural-Controllers for the Game of Predator vs. Prey

Adam Balint

Supervised by Beatrice Ombuki-Berman

Submitted in partial fulfillment
of the requirements for COSC 4F90

Department of Computer Science

Brock University
St. Catharines, Ontario

Abstract

This thesis examines the emerged behaviour with PSO trained neural controllers for the problem of predator vs prey. Variations of game rules were tried in order to counteract undesirable traits encourage development of the creatures. This included variations such as the agents being able to fall off the board, and various scoring systems for the outcome of the game. Although initially some promising simplistic behaviour was seen, and as the game rules were developed some more complex and surprising behaviour emerged. The prey always emerged victorious with some clever tactics being evolved to outsmart the predators and take advantage of the game rules.

Contents

List of Tables

List of Figures

1 Introduction

This thesis focuses on the competitive co-evolution of neural controllers with the hopes of evolving complex behaviour in the game of predator vs prey. Played on a board sized 9x9, the agents move simultaneously in hopes that the predator captures the prey while the prey attempts to avoid the predator. To encourage evolution within the system, walled and non-walled boards were tried along with three different variations in game rules. These game rules were adapted as the system was tested in order to counteract undesirable developments in previous versions of the game, or to encourage certain behaviours in the new version. To much surprise, the prey always emerged victorious by developing some interesting strategies. Some of these strategies involved running off the board in a walled off game, exploiting the game rules in their favour, or adapting to the predator tactics in order to emerge victorious. When facing these advanced prey, the predators did not stand much of a chance, as such further system improvements are suggested in order to even the playing field.

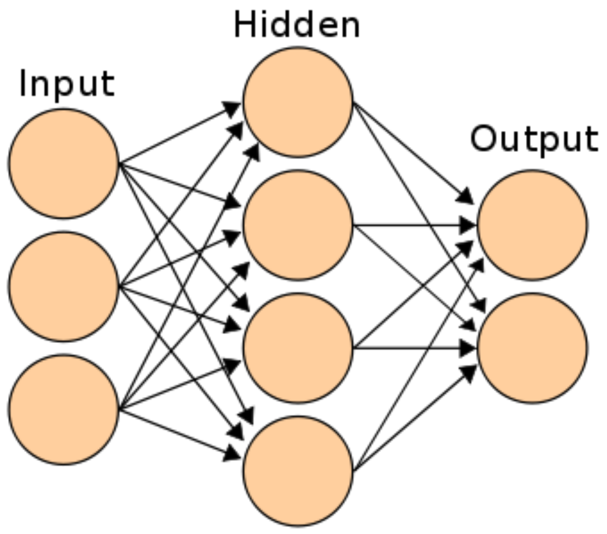


Figure 1: Representation of a neural network[9]

2 Background

2.1 Artificial Neural Networks

Neural networks are a type of computational model that is inspired by the way brains function [5][6]. Artificial neural networks are made up of component nodes which are stacked into layers and connected to form a network (see Figure 1). The neural network operation consists of providing input data to the input layer of the network, passing that data along the network to the output layer and then interpreting the output in a specific way. In the case of predator vs prey, output neurons will correspond to different actions that the predator and prey can make.

2.1.1 Neurons

The key component nodes of a neural network are called neurons. The sole job of the neuron is to compute the weighted sum of all of the inputs to the neuron which could be from the input data or from other neurons. The computed value is then passed through an activation function. The layout of a neuron can be visualized in Figure 2.

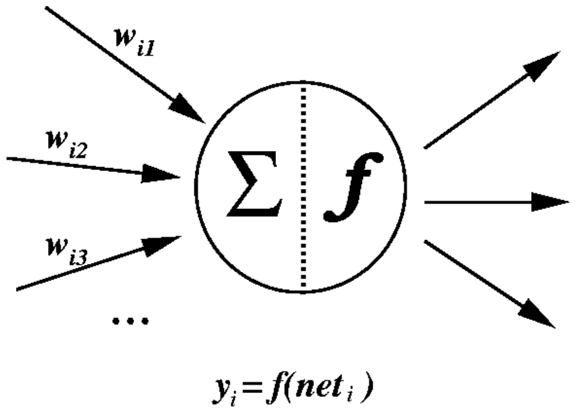
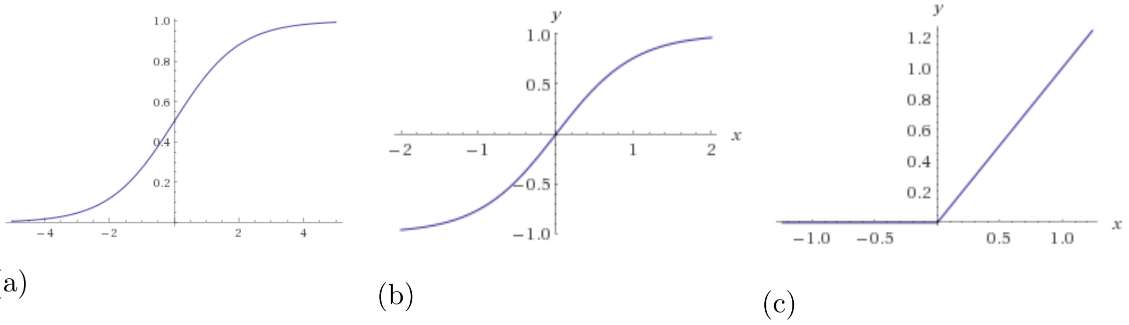


Figure 2: Representation of a single neuron in a neural network[8]



$$y = \frac{1}{1 + e^{(-x)}}$$

logistic function

$$y = tanh(x)$$

tanh function

$$y = f(x) = max(0, x)$$

rectifier function

Figure 3: Common Activation Functions

2.1.2 Activation Functions

In neural network literature there are several common activation functions that used[6]. The three most used ones are the logistic function, tanh function and rectifier function. These functions can be seen in Figure 3 along with their respective formulas. For the purposes of this thesis all activation functions used are the logistic function.

2.2 Particle Swarm Optimization

2.2.1 Base PSO Overview

Particle swarm optimization(PSO) was proposed by [10] and it takes inspiration from the flocking behaviour of swarms of birds. When observing birds there tends to be one leader and the rest of the flock tend to follow that leader. Particles which are analogous to birds are spread throughout a continuous solution space where the particle with the best solution to the problem becomes the leader (global best). Throughout multiple generational time steps the rest of the particles converge towards the global best. If a new best solution is found, the particle that found the new best solution becomes the global best and the process continues until convergence or a max generation cap is reached. This technique allows for a spread out simultaneous search over the search space.

2.2.2 Particle

The particle is the base unit of the PSO. It contains a position within the search space and a velocity. In addition, particles also keep track of their best position until the current time step. This location is the location that has provided the most optimal solution thus far. In the beginning of the algorithm, both the position and the velocity are randomized. This allows the particles to be spread throughout the search space. As the PSO algorithm runs, the particles will move towards the best location, searching the search space along the way.

Each particle also contains a social and cognitive coefficient along with an inertial coefficient that influence how the particle behaves. The social coefficient modifies the influence of the global best position, while the cognitive coefficient modifies the influence of the personal best location. The inertial coefficient modifies the amount of the previous velocity that is carried over to the next time step. These coefficients are used in the velocity update formula, and tend to be picked according to the graph in Figure 4 that was proven to be convergent by [1].

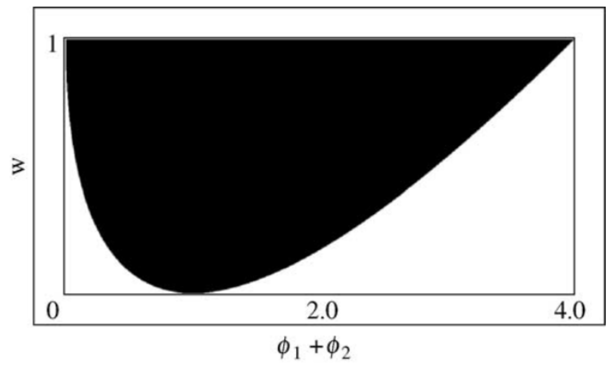


Figure 4: Parameter convergence graph for PSO[1]

2.2.3 Particle Position Update

The particle position update can be considered as a simplified representation of the concept of motion in classical mechanics. Since each particle contains a velocity vector that represents the direction and magnitude that the particle will move in, that velocity is added to the current position of the particle causing it to move to a different location. The formula is as follows:

$$x(t+1) = x(t) + v(t) \tag{1}$$

where:

x represents the position of the particle

v represents the velocity of the particle

2.2.4 Particle Velocity Update

Each time step, the velocity of the particle is also updated to take into account the particle's current location. In each time step as particles move, the particle's personal best location and the swarm's global best solution could have changed. Therefore, the velocity update formula must take this into account. Another concept that is utilized in the velocity update formula is the idea of inertia from physics. If an object is moving in a certain direction, the object would like to keep moving in that direction unless acted on by another force. This means that if a different force is applied, then the velocity will become the sum of the two forces. When visualized, this causes a smother motion to the particles allowing them to cover more of the search space. The velocity update formula follows below:

$$v(t+1) = \omega v(t) + c_1\psi_1(p(t) - x(t)) + c_2\psi_2(g(t) - x(t)) \tag{2}$$

where:

v is the velocity of the particle

ω is the inertia coefficient

c_1 and c_2 are the personal and social coefficients respectively

ψ is a random multiplier between 0 and 1

$x(t)$ is the location of the particle at time t

$p(t)$ is the personal best location of the particle at time t

$g(t)$ is the global best location of the swarm at time t

2.2.5 Particle Evaluation

Particle evaluation is problem dependent. For general benchmarking and testing, functions such as the Ackley function and Eggholder function are used to calculate the fitness. The evaluation function for the particles that are used for the predator vs prey problem will be described in-depth in section 4.3.2.

2.2.6 Swarms

Swarms are the major force behind the PSO. A swarm consists of a population of particles. Each swarm is responsible for maintaining the global best position among all of its particles. This is the same value that is used in the velocity update formula. The PSO algorithm could make use of multiple swarms when using algorithms such as the Co-operative Particle Swarm Optimization or Competitive Particle Swarm Optimization algorithms.

The pseudocode for the PSO algorithm can be seen in Figure 5. As it can be seen, the algorithm begins with initializing a population of particles with random locations and velocities. The swarm best is calculated. The algorithm runs until a specified stop condition. In this thesis, this stop condition was reached after 500 iterations of the location update. The algorithm then updates the velocities and positions of every particle before updating the personal best and global best locations.

2.3 Charged PSO

2.3.1 Charged PSO Overview

Charged PSO is a variation on the normal PSO algorithm[11]. This variation introduces charged particles in order to prevent full convergence and to reduce convergence speed in general. The original PSO algorithm works well for static environments, however it falls short in dynamic environments. As the charged PSO does not fully converge, it is more suitable for dynamic environments with constantly changing error landscapes.

2.3.2 Charged Particles and Updating

The introduction of charged particles is the defining feature of charged PSO. Similarly to the original PSO particle, charged particles also contain a position, velocity and personal best location, however they also carry a charge. This charge can differ from particle to particle.

The charged particle's position update formula is the sum of the velocity and the particle's current location. This formula is identical to the normal PSO's particle position update equation.

The charged particle's velocity update formula introduces an additional term to the formula seen in the normal particle velocity update formula. The new velocity update formula is as follows:

$$v(t+1) = \omega v(t) + c_1\psi_1(p(t) - x(t)) + c_2\psi_2(g(t) - x(t)) + a_i(t) \tag{3}$$

This introduces the term $a_i(t)$ to Formula 2 which a force that is calculated for each particle and is the repulsion force that a charged particle feels from other charged particles in the area. The repulsion force is given by the formula:

$$a_i(t) = \sum_{j=1, j \neq i}^{n_s} a_{ij}(t) \tag{4}$$

where:

$$a_{ij} = \begin{cases} \frac{Q_i Q_j}{d_{ij}^3} * (x_i(t) - x_j(t)) & \text{if } R_c \leq d_{ij} \leq R_p \\ \frac{Q_i Q_j}{R_c^2 d_{ij}} * (x_i(t) - x_j(t)) & \text{if } R_c < d_{ij} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where:

$d_{ij} = ||(x_i(t) - x_j(t))||$ or the Euclidian distance between particles i and j

Q is the charge of the particle

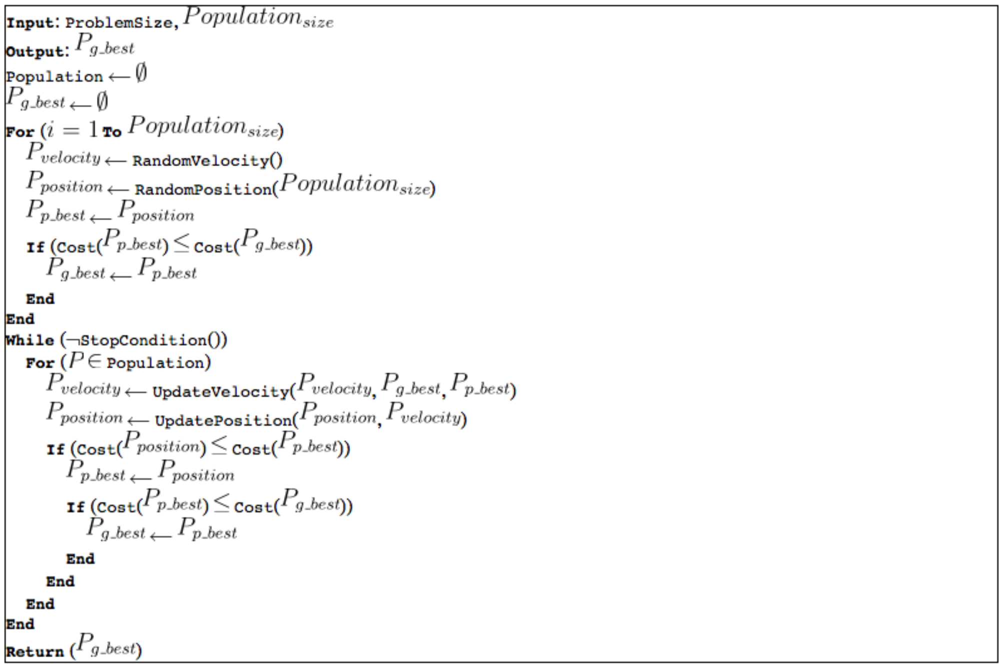
R_c is the core radius

R_p is the particle perception limit

This equation can be imagined as a particle surrounded by two fields. If a charged particle is outside the outer field (provided by the perception limit) then the particles are far enough apart from each other that no force is experienced between the particles. Within the perception limit but outside the core radius, the force experienced between the particles scales with the distance. The closer the particles are the more of a repulsion force is experienced. Finally, if the particle is within the core radius limit, the maximum repulsive force is experienced by both particles. This repulsive force ensures that the particle swarm never fully converges. This prevention of full convergence ensures that the PSO can constantly improve in fitness even in dynamic environments.

2.3.3 Charged Swarms

Similarly to normal swarms, charged swarms are responsible for the maintenance of the global best position of all the particles in the swarm. Charged swarms can be made up of



```
Input: ProblemSize, $Population_{size}$
Output: $P_{g\_best}$
Population $\longleftarrow \emptyset$
$P_{g\_best} \longleftarrow \emptyset$
For ($i = 1$ To $Population_{size}$)
  $P_{velocity} \longleftarrow$ RandomVelocity()
  $P_{position} \longleftarrow$ RandomPosition($Population_{size}$)
  $P_{p\_best} \longleftarrow P_{position}$
  If (Cost($P_{p\_best}$) $\leq$ Cost($P_{g\_best}$))
    $P_{g\_best} \longleftarrow P_{p\_best}$
  End
End
While ($\neg$StopCondition())
  For ($P \in$ Population)
    $P_{velocity} \longleftarrow$ UpdateVelocity($P_{velocity}$, $P_{g\_best}$, $P_{p\_best}$)
    $P_{position} \longleftarrow$ UpdatePosition($P_{position}$, $P_{velocity}$)
    If (Cost($P_{position}$) $\leq$ Cost($P_{p\_best}$))
      $P_{p\_best} \longleftarrow P_{position}$
      If (Cost($P_{p\_best}$) $\leq$ Cost($P_{g\_best}$))
        $P_{g\_best} \longleftarrow P_{p\_best}$
      End
    End
  End
End
Return ($P_{g\_best}$)
```

Figure 5: PSO Pseudocode[7]

a combination of charged and normal(neutral) particles. As the charged particles interact and repel each other, the charged particles explore the changing landscape searching for better solutions, while the neutral particles provide a more concentrated search around the global best. Since the swarm is also aware of each particle, the charged swarm may also be accountable for calculation of the repulsion force experienced by each particle.

3 Literature Review

3.1 Cooperative Coevolution

Cooperative coevolution can be defined as the evolution of multiple agents that coordinate behaviour in order to improve together and achieve a common goal[15]. A real life example could be the organization of an ant colony. In the colony, ants have different roles, but all of the ants work together to ensure that the colony as a whole survives. Another example where cooperation is necessary is in emergency response situations[16][17]. In emergency situations, the fire department, emergency medical services and the police department need to work in tandem in order to effectively service the area.

3.2 Competitive Coevolution

Competitive coevolution on the other hand can be seen in games [12][13][14]. As an example consider two people who are playing a game such as chess or go. If one player begins to play better, then the other player will adapt and reaching the same level if not surpassing the other player's level. Another example of competitive coevolution is the RoboCup and the Federation of International Robot-soccer Association (FIRA)[3]. These two organizations hold robot soccer competitions, both real and virtual with both complete and incomplete information of the system. Competitive coevolution in this case would be the two teams of robots improving their techniques in order to counter the other team.

3.3 Neural Network Training

3.3.1 Backpropagation

Backpropagation is the conventional way to train neural networks [20]. This technique utilizes the error at the output nodes and uses the partial gradients of the weights in order to assign a portion of the error to each node according to that node's contribution to the error. A portion is this error is then applied to update the weights. Since backpropagation tends to rely on a constant error landscape, applying basic backpropagation to a dynamic environment is not logical.

3.3.2 Genetic Algorithms

Genetic algorithms have successfully been applied to training neural networks[19] with approaches such as the Evolving Neural Networks through Augmented Topologies (NEAT) algorithm [18]. As genetic algorithms are well suited for discrete domains, these methods tend to focus on evolving the structure of the neural network rather than evolving the value of the weights.

3.3.3 PSO

PSO had been applied to training neural networks for classification by [21], although they were not as good as the author was expecting. Because basic PSO tends to converge fully, base PSO can be easily used to train neural networks in static environments. However, basic PSO would not be suitable for a dynamic environment like the predator vs prey game. As such, charged PSO was successfully used to evolve neural networks for the predator vs prey game[2].

3.4 Predator vs. Prey

The type of competitive coevolution that this thesis focuses on is the predator vs prey game. This game is similar to the robot soccer in the way that both the predator and the prey attempt to out maneuver each other in order to emerge victorious. This thesis implements and expands on a variation of [2] on evolving behaviour through competition for the predator vs prey game. In [2], the predator vs prey game was implemented using a closed board. There were 50 prey and 50 predators that played a total of 800 turn-based games, 400 of which the predator made the first move and 400 of which the prey made the first move. Langenhoven's implementation of the predators and prey allowed for both agents to take the following actions: move along x-axis, move along y-axis, do not move. The prey has an additional option of jumping in which case the prey moves two squares rather than one. The inputs for both the predators and the prey were the location of the agent and the opponent scaled to within the range of the hyperbolic tangent function that was used.

The main differences between the system proposed in [2] and the system used in this thesis is the dimensionality of the input and the fact that the games are based on simultaneous movement rather than an action-reaction based turn taking system. The system used in this thesis is discussed in section 4.

4 Experimental Setup

4.1 Predator vs Prey

The predator vs prey game consists of two types of agents (predators and prey) competing on a nine by nine board. The goal of the predator is to capture the prey and the goal of the prey is to avoid the predator for the 20 turns. Both the predator and prey move simultaneously in this implementation. This should cause both the predators and the prey to learn to predict the movement patterns of their opponent .

4.1.1 Predator

As stated before, the goal of the predator is to catch the prey. In order to interact with the environment, the predator has a neural network that is associated with it. This neural network's input layer consists of 81 input nodes which read the 9 by 9 board state. There are a certain number of hidden nodes that were varied between runs, and there are 7 output nodes. The output node actions are arranged as seen in Table 1. Some output nodes in the neural network are connected. These nodes are listed together in the table, and when the action for the set is decided, the node with the higher activation is used. The predator has two unique output nodes that the prey does not have access to. These output nodes apply a two times multiplier to the movement in either the horizontal or vertical direction. This could be thought of similar to a jump or pounce action from the predator's point of view. If there is no diagonal movement, then the largest of the 2, 3, 4 and 5 nodes will be taken to determine the direction of the movement.

4.1.2 Prey

The actions of the prey are set up similarly to the actions of the predator. However, the prey have a slightly smaller neural network. The prey neural network consists of the same 81 input nodes which accept the board state, followed by a certain number of hidden nodes that

Output Node	Action
1	Move diagonally (yes or no)
2 and 3	Horizontal movement left or right
4 and 5	Vertical movement up or down
6	Chase horizontally
7	Chase vertically

Table 1: Predator output node to action mapping

Output Node	Action
1	Move diagonally (yes or no)
2 and 3	Horizontal movement left or right
4 and 5	Vertical movement up or down

Table 2: Prey output node to action mapping

were varied between runs. Unlike the predators, they only have 5 output nodes. The actions that the prey can perform are summarized in Table 2. The prey's neural network does not contain the special nodes used by the predator. Once again, node 1 indicates whether or not the prey will mode diagonally. The node set of 2 and 3, and set 4 and 5, the node with the higher activation is taken. If there is no diagonal movement, then the largest of the 2, 3, 4 and 5 nodes will be taken to determine the direction of the movement.

4.1.3 Game Progression

As the game is played and the learning of both the predator and prey progresses, some large gains and falls in fitness should be observed as one type of agent outsmarts the other and vice versa. The runs are expected to reach an equilibrium when averaged over time.

4.2 NN Implementation

On an implementation level, the neural networks were implemented using an array of matrices which are multiplied together to calculate the output of the network. Each matrix represents the connection weights between 2 layers of the neural network. To accomplish this matrix representation, the RealMatrix library provided by [4] was used. The activation function utilized by all the nodes in the neural network was the logistic function. The weights of the connections (represented by the matrix) was evolved using particle swarm optimization.

4.3 PSO Implementation

4.3.1 Particle Representation

As the PSO is used to evolve the neural network, a particle representation of the neural network weights must be created. This was accomplished by using the weight values in the neural network matrices as the position of the particle. This results in a high dimensional particle where the dimensionality of the particle is equal to the number of connections in the neural network. Since the representation of the weights in the neural network is one-to-one to the location of the particle, as the particle location changes and updates the weights of the network are also updated.

4.3.2 Calculating Fitness Values

Fitness values for particles are calculated by instructing each particle to play 400 games against particles from the other swarm. There were three different point systems that were experimented with. These are outlined in Table 3. The fitness value calculated was divided by the number of games that was played by the particle in order to reduce the fitness to a range between -2 and +1. As seen in Table 3, game V1 is heavily stacked in favour of the prey, as if the prey or predator fall off then the predator does not capture the prey and the prey gains a point. Game V2 is a balanced match between the two types of agents while game V3 provides a slight advantage to the prey because if the prey falls off then the predator loses one point, while if the predator falls off then the prey gains one point as it successfully avoided the predator.

Game Version	Scenario	Predator Score	Prey Score
V1	Predator catches prey	+1	-1
	Prey avoids predator	-1	+1
	Predator and prey fall	0	+1
	Predator falls	0	+1
	Prey falls	0	+1
V2	Predator catches prey	+1	-1
	Prey avoids predator	-1	+1
	Predator and prey fall	-2	-2
	Predator falls	-2	0
	Prey falls	0	-2
V3	Predator catches prey	+1	-1
	Prey avoids predator	-1	+1
	Predator and prey fall	-2	-2
	Predator falls, prey does not fall	-2	+1
	Predator does not fall, prey falls	-1	-2

Table 3: Game Rules

4.3.3 Simulation

To set up the simulation for a particle, an opponent is selected randomly from the opposing swarm. The predator and prey are then both placed on the board randomly such that the predator is on the upper half of the board and the prey is on the lower half of the board. Each turn consists of the predator and prey reading in the board state and then deciding which move to make. The predator and prey then simultaneously make their moves for 20 turns. The game ends if the predator catches the prey, or if the 20 turns have been used.

4.3.4 Board Encoding

As mentioned, the neural network takes the board as input. Therefore, the board has to be encoded before it is given to the network. This board encoding encodes the 9x9 board as seen in Table 4.

Board State	Encoding
Empty Square	0
Same Species	1
Opposite Species	-1

Table 4: Game Board Encodings

4.3.5 Hall of Fame

Both the predator and prey swarm have their own hall of fame which stores prior global best values of that swarm in order to provide some backtracking over time to see if a previous set of weights is more efficient than the current best weights. Each hall of fame can hold a maximum of 25 fitness values. The fitness values stored were the last 25 global best fitness values. The hall of fame is updated every time a new global best fitness is determined and the new global best weight value is stored in the hall of fame. The values in the hall of fame are re-run every 10 epochs in order to determine if any older values have a better fitness than the current best as these could change as the opponent swarm improves.

4.4 Visual Board Representation

In order to visualize the games played by the predator and prey agents, a visual board representation was developed. The representation consists of a nine by nine grid with two agents placed on it. The predator agent is represented by a red circle, while the prey agent is represented by a green circle. The starting location of both agents are left unmarked, however a line is drawn between the old location and the next location that the agent moved to. Therefore, the tile with a line connecting to it without a circle is the starting tile. A labeled example of a game can be seen in Figure 6.

4.5 Run Summary

Table 5 contains all of the parameter sets that were tested. Throughout the experiments, the percentage of charged particles, core radius and perception limit combinations, and number

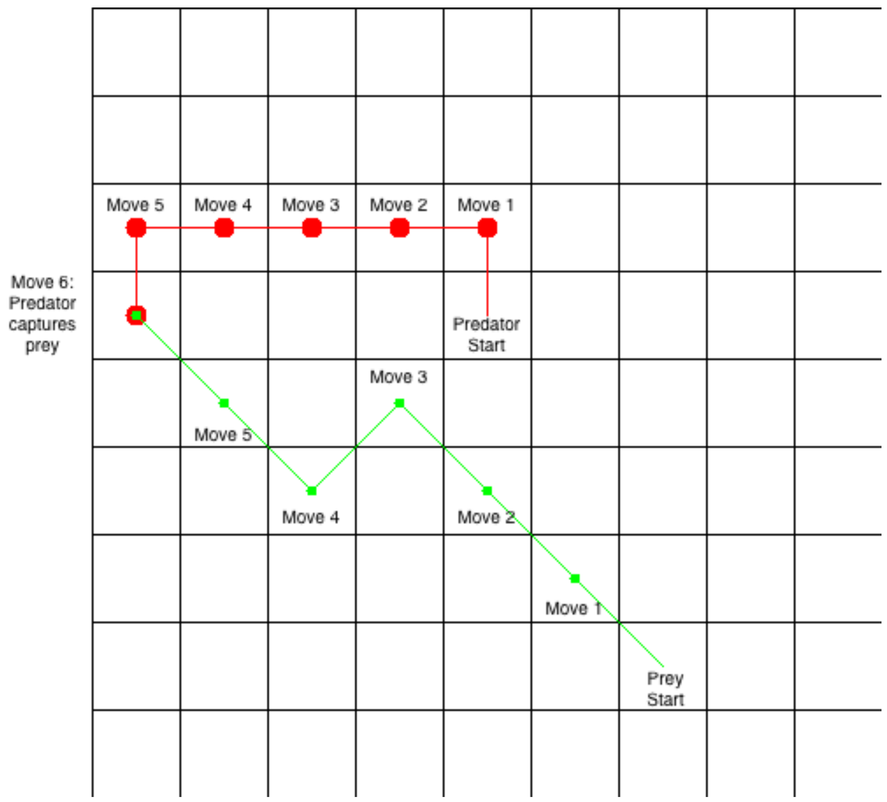


Figure 6: Labeled visualization of the game board

of hidden nodes were varied. The ability to fall off the board was also changed in order to see if any behavioural changes would develop. All charged particles had a charge value of 16. If a closer look at the game rules is taken, then it can be seen that if the ability to fall off is removed, then all three game rules simplify to the same values. In other words, if the predator catches the prey, then the predator gains one point while the prey loses a point. If the prey avoids the predator, then the opposite takes place. When discussing the results, the game rules version number will be prepended to the experiment number excluding when the agents cannot fall off the board. For each of the parameter sets, 5 runs were conducted. Each run consisted of 50 predators and 50 prey grouped into two swarms, one per agent type. The swarms were evolved over 500 iterations. Each game that was played took place over 20 turns unless the predator caught the prey, or either the predator or the prey fell off of the board.

Experiment	Hidden Nodes	Core Radius	Perception Limit	Swarm Charged (%)
1	15	2	40	0
2				33
3				67
4				100
5		5	30	0
6				33
7				67
8				100
9	30	2	40	0
10				33
11				67
12				100
13		5	30	0
14				33
15				67
16				100
17	60	2	40	0
18				33
19				67
20				100
21		5	30	0
22				33
23				67
24				100

Table 5: Experiment Summary

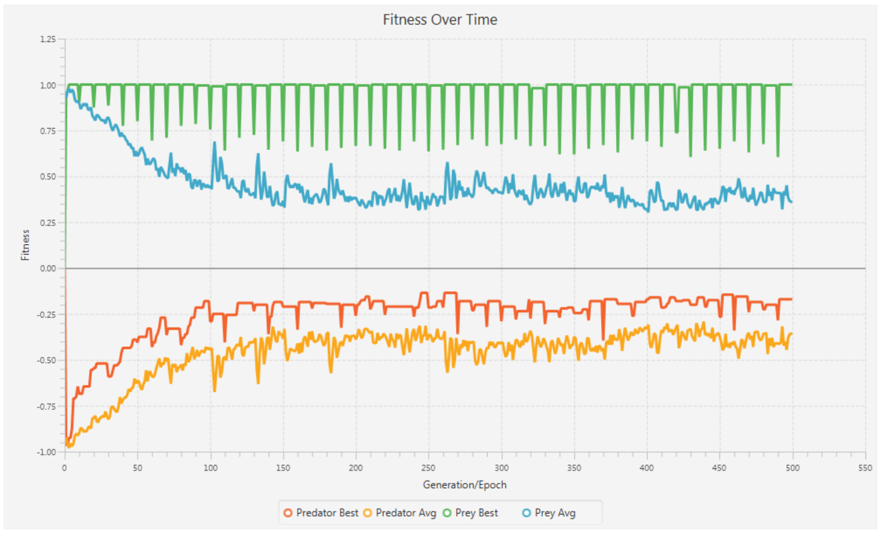
5 Results

The results section will explain the results that were obtained using the parameter sets and will demonstrate any evolved behaviour. The walled game encompasses all of the experiments in Table 5 using the rules in Table 3 on a walled off board. The Game V1, V2 and V3 sections will discuss all of the experiments on a board where the agent can fall off using the corresponding V1, V2, and V3 rules.

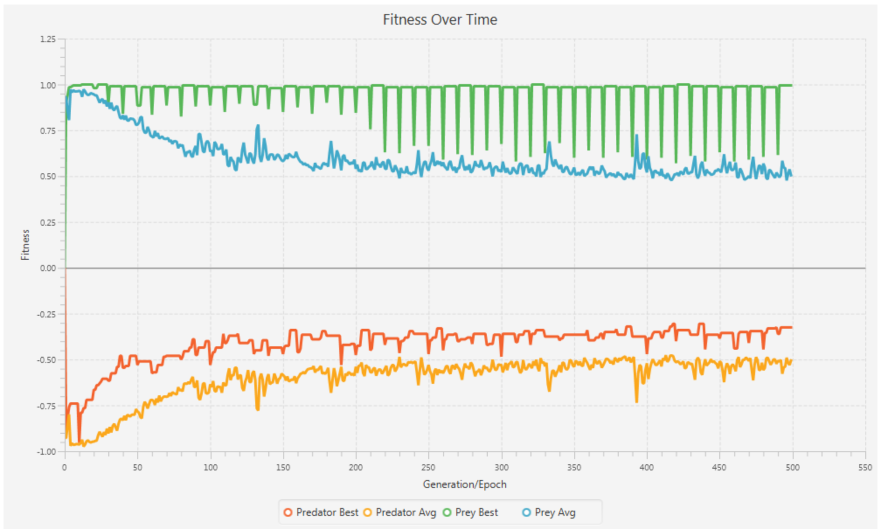
5.1 Walled Game

When the arena is walled off and the agents are not able to fall off of the board, it is hypothesized that the prey will figure out some sort of move pattern to avoid the predator quickly, and the predator should be able to quickly adapt to the prey and capture it. This would show on fitness graphs as lines representing the predators and prey crossing over each other as they keep adapting to capture or escape respectively. This type of behaviour is not seen in any of the walled off game graphs, although the fitnesses do fluctuate in tandem with each other, just not as extremely as expected. Furthermore, the prey always outperforms the predator in the walled off game. The general pattern of all of the graphs tend to be a sharp improvement in the beginning of the run, and then the curve flattens out as seen in Figure 7 in which both runs came from the same experiment. In most cases the fitness of the particles at the end of the run are determined by the gain or loss that occurred at the beginning of the run. A more promising run that seems like it would have improved if training time was longer can be seen in Figure 8.

In the walled game, although the overall population performance for the predator is not great, there tends to be at least one strong particle that is much higher than the rest of the swarm. This can be seen in Figures 10 and 11. An example of the types of games this type of particle plays can be seen in Figure 9. In this case, the prey always seems to want to leave the stage. However, since it cannot, it seems to get stuck, and then the predator just



(a) Run 0



(b) Run 2

Figure 7: Experiment 13 fitness over time

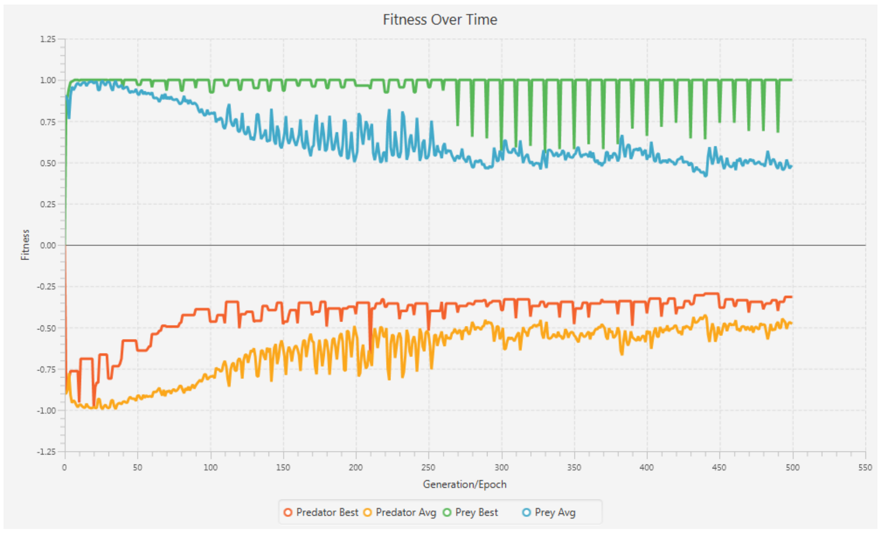


Figure 8: Experiment 1 run 2 fitness over time

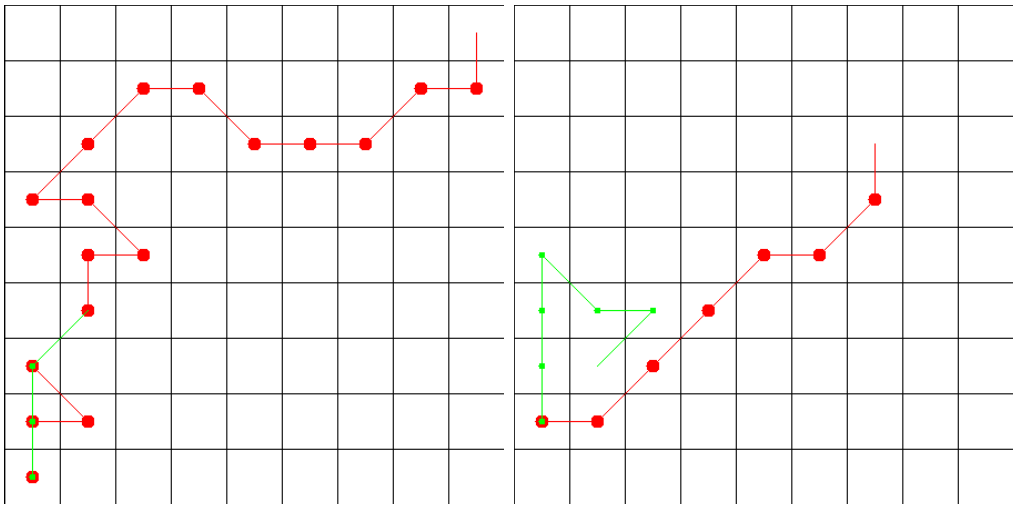


Figure 9: Experiment 15 run 2: Example games

approaches and captures the prey. Although this is a very simple scenario, the path of the predator is quite direct when going to capture the prey, as such it demonstrates that the predator is capable of learning to capture the prey.

Since only 5 runs were conducted per experiment, the results are not statistically significant, however, in general, both the predator and the prey perform better using the core radius of 5 and perception limit of 30. Furthermore, the predator performs better with a smaller network, while network size has no impact on the prey results. This implies that for the walled off game, a smaller network is enough for the predator to predict the movements

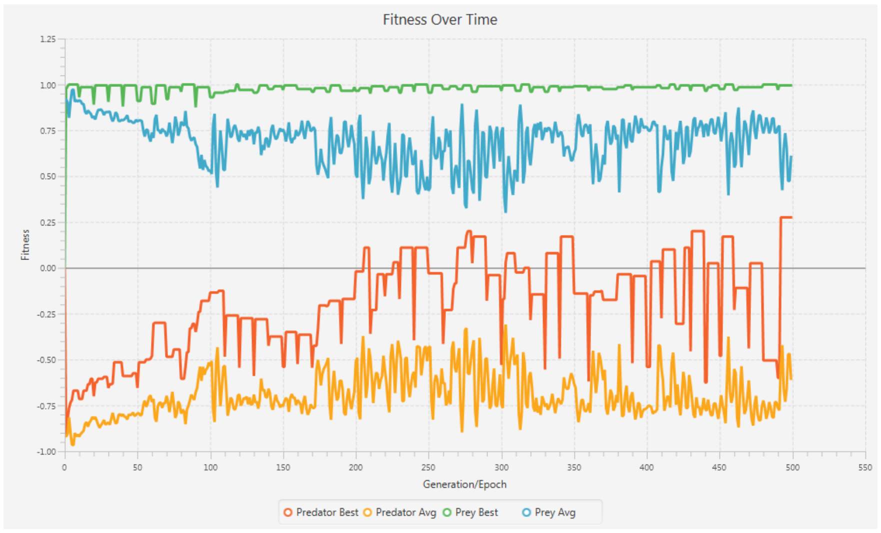


Figure 10: Experiment 15 run 2 fitness over time

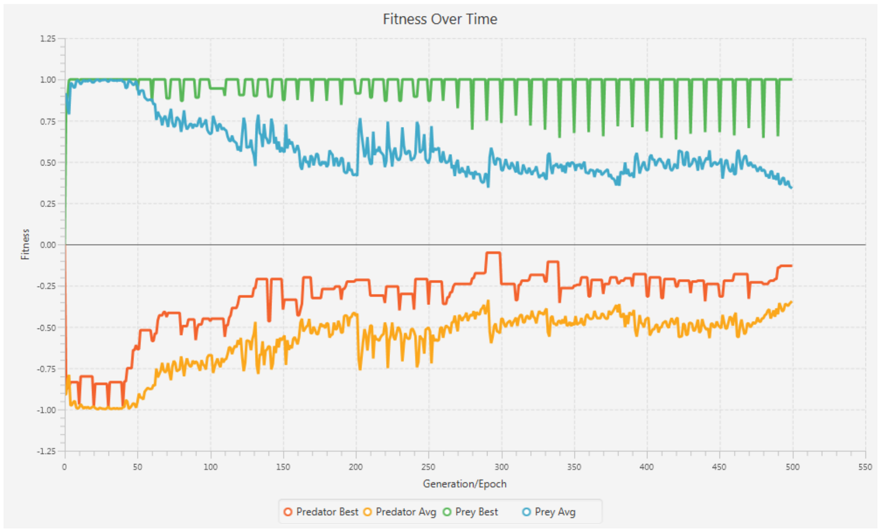


Figure 11: Experiment 6 run 1 fitness over time

Experiment	Predator			Prey		
	Max	Average	Min	Max	Average	Min
1	-0.235	-0.363	-0.475	0.995	0.964	0.943
2	0.065	-0.245	-0.455	0.980	0.965	0.942
3	-0.055	-0.278	-0.465	0.994	0.973	0.940
4	-0.125	-0.271	-0.460	0.988	0.969	0.939
5	-0.200	-0.312	-0.405	0.997	0.960	0.918
6	-0.050	-0.253	-0.450	0.999	0.993	0.984
7	-0.075	-0.291	-0.440	0.981	0.953	0.922
8	0.025	-0.160	-0.415	0.992	0.973	0.944
9	-0.135	-0.284	-0.415	0.984	0.974	0.964
10	-0.305	-0.392	-0.430	0.995	0.963	0.914
11	-0.125	-0.307	-0.430	0.991	0.966	0.911
12	-0.230	-0.309	-0.340	0.989	0.969	0.937
13	-0.135	-0.355	-0.500	0.977	0.961	0.933
14	-0.145	-0.355	-0.505	0.967	0.955	0.937
15	0.275	-0.131	-0.370	0.985	0.974	0.957
16	-0.195	-0.310	-0.385	0.994	0.980	0.957
17	-0.260	-0.358	-0.540	0.996	0.979	0.966
18	-0.285	-0.392	-0.445	0.998	0.963	0.926
19	0.015	-0.213	-0.430	0.988	0.979	0.968
20	-0.195	-0.343	-0.465	0.988	0.947	0.918
21	-0.040	-0.297	-0.420	0.988	0.960	0.946
22	-0.125	-0.273	-0.440	0.994	0.973	0.943
23	0.005	-0.296	-0.555	0.993	0.969	0.939
24	-0.040	-0.330	-0.455	0.994	0.961	0.921

Table 6: Walled Game Summary

of and capture the prey.

5.2 Game V1

As seen in Table 3, the game V1 rules are heavily stacked in favour of the prey. The effect of this can be seen in all of the fitness over time graphs that were produced. Even the experiments that included charged particles performed poorly with this version of the game. An example of a graph from experiments 1 and 17 can be seen in Figure 12 and an example of a graph from experiment 4 and 20 can be seen in Figure 13. In Figure 13 it is expected that at least one particle would have a high fitness value, but as it can be seen, the full charged swarms from experiment 4 and 20 do not perform better than those from experiments 1 and 17.

When observing the games being played by the particle with the best fitness over all of the epochs an interesting pattern emerges. In most cases, the game ends with the prey running off of the board. This is because the easiest way for the prey to win is for it to fall off. While the predator also falls off about one third of the time, the predator tends to stay on the board for a longer time than the prey. An example of this technique can be seen in Figure 14. The falling off behaviour of the prey can be reinforced by the fact that the best prey had 324 games of 400 ending with the prey falling off the board. An example of several games can be seen in Figure 15. Unfortunately, since it is extremely easy for the prey to win and for the predator to lose, the type of behaviour that evolved is quite basic. In general, both agents tend to just move in a straight line until one of them falls off the board.

In Table 7, it can be seen that the parameter set did not provide much change in the prey results, while the highest values produced by the predator all occurred when 33% of the swarm was charged. This result does seem promising, however the core radius and perception limit parameters are not the same. Observing the core radius and perception limit, the set with the lower perception limit on average provides better fitness scores. Furthermore, for larger networks, the core radius of 5 and perception limit of 30 seems to perform better than

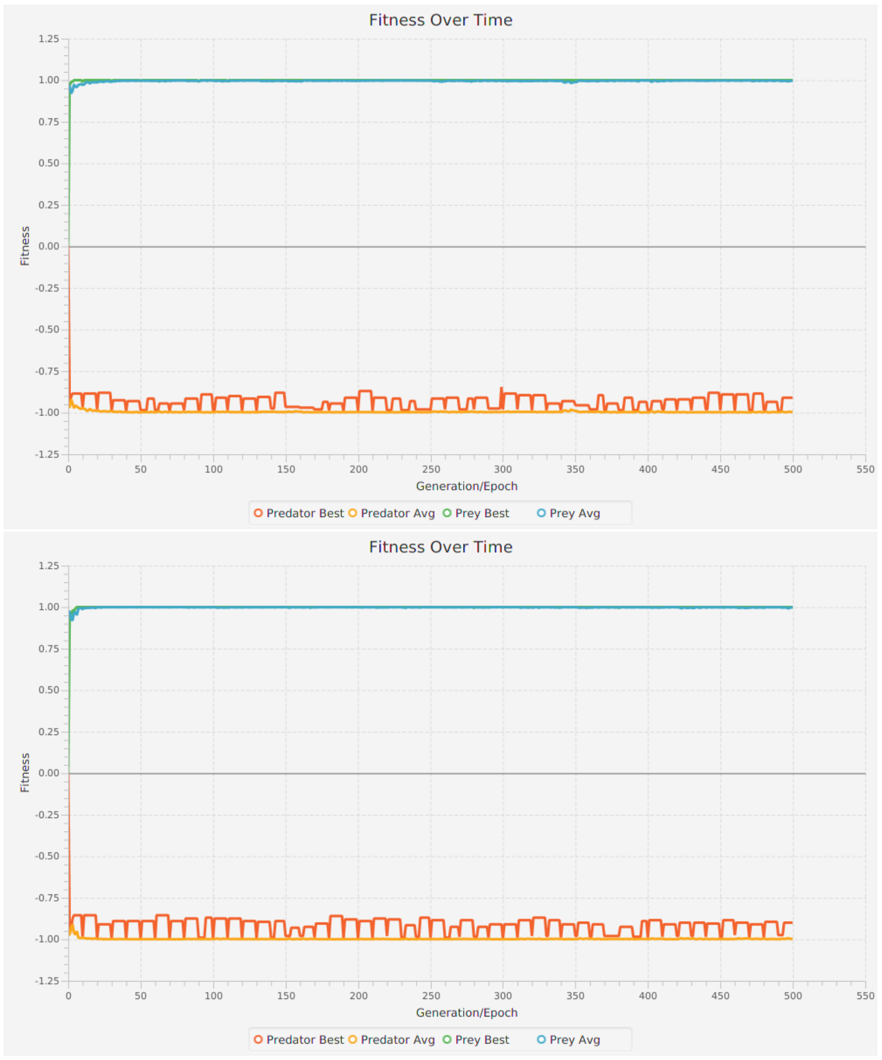


Figure 12: Game V1 Experiments 1(top) and 17(bottom)

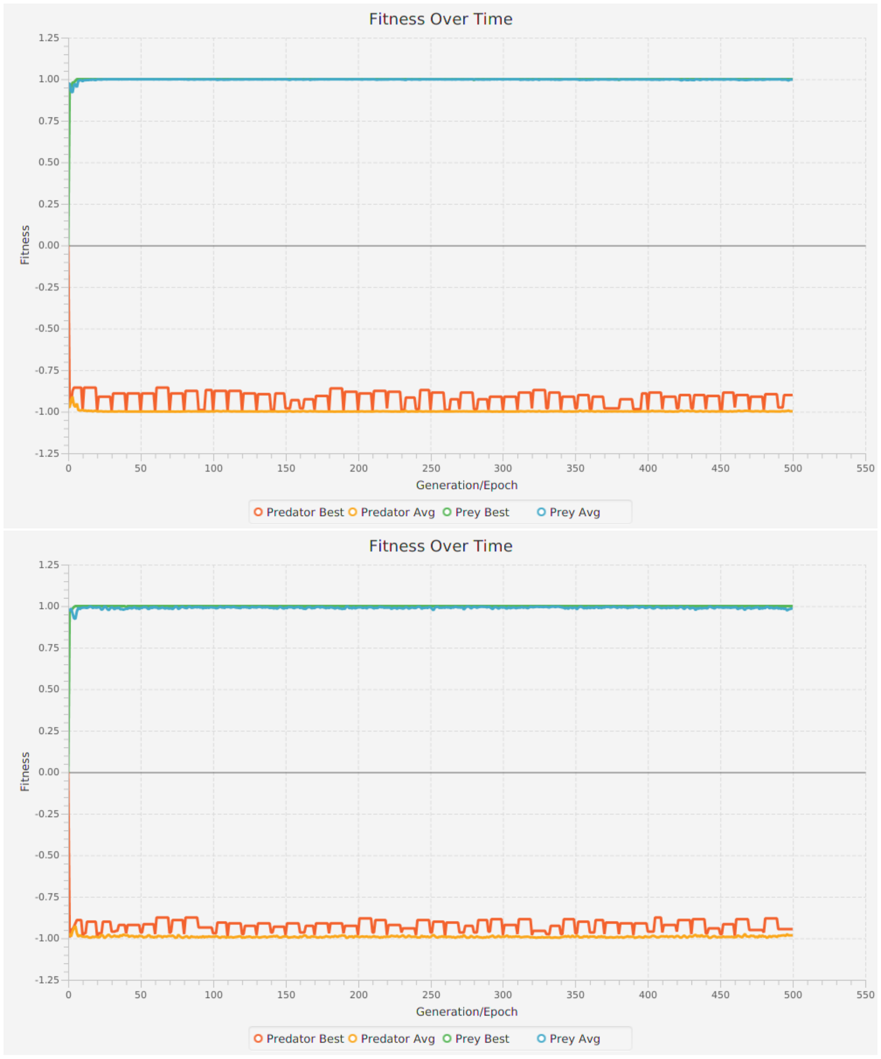


Figure 13: Game V1 Experiments 4(top) and 21(bottom)

Experiment	Predator			Prey		
	Max	Average	Min	Max	Average	Min
1	-0.845	-0.875	-0.900	1.000	1.000	0.999
2	-0.815	-0.854	-0.875	1.000	0.999	0.997
3	-0.845	-0.870	-0.890	1.000	0.998	0.996
4	-0.860	-0.879	-0.915	1.000	0.998	0.996
5	-0.835	-0.868	-0.920	1.000	1.000	0.998
6	-0.850	-0.868	-0.905	1.000	0.999	0.997
7	-0.850	-0.876	-0.915	1.000	0.998	0.995
8	-0.885	-0.898	-0.905	1.000	0.997	0.992
9	-0.865	-0.884	-0.905	1.000	0.999	0.998
10	-0.800	-0.874	-0.905	1.000	0.999	0.998
11	-0.840	-0.867	-0.895	1.000	0.997	0.995
12	-0.870	-0.877	-0.885	0.998	0.996	0.994
13	-0.850	-0.871	-0.905	1.000	0.999	0.998
14	-0.820	-0.877	-0.900	1.000	1.000	1.000
15	-0.835	-0.877	-0.900	1.000	0.999	0.997
16	-0.850	-0.888	-0.910	1.000	0.999	0.995
17	-0.855	-0.875	-0.915	1.000	1.000	0.999
18	-0.855	-0.890	-0.910	1.000	1.000	0.999
19	-0.840	-0.869	-0.905	1.000	0.999	0.998
20	-0.855	-0.882	-0.905	0.999	0.998	0.996
21	-0.845	-0.874	-0.900	1.000	1.000	1.000
22	-0.790	-0.830	-0.895	1.000	0.999	0.998
23	-0.815	-0.854	-0.880	1.000	0.999	0.999
24	-0.810	-0.882	-0.920	1.000	0.998	0.995

Table 7: Game V1 Summary

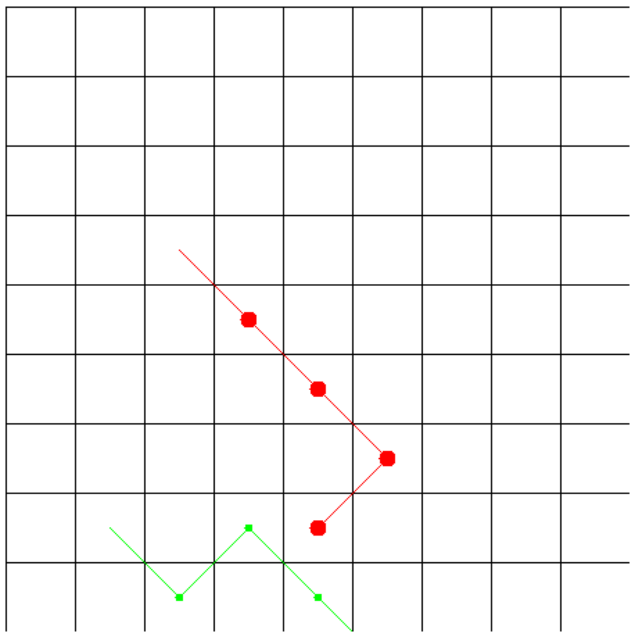


Figure 14: Game V1 Experiment 1: Example predator game

the other variation when all other parameters are identical.

5.3 Game V2

Game V2 is the game with the even set of rules for the predators and the prey. This game mode should be the mode that provides the closest set of results between the predators and the prey. Unfortunately, the system still seems to provide an advantage to the prey as in every experiment the prey are the agents with the highest fitness. However, unlike in the walled off game and the V1 game, in this game mode, there are times when the predators have a better score than the prey (Figures 16b and 17). Unfortunately for the predator, they always get outperformed by the prey. It should be noted that in this case, a core radius of 5 and a perception limit of 30 does make an impact in the initial fitness of the particles, and it is in these cases where the predator fitness is initially higher than the prey fitness.

Something of note is that in this game mode, the prey seem to converge as in some games, even when the particles are 67% charged, the prey experience large drops in fitness when the predators gain a small amount. This can be seen in Figure 18.

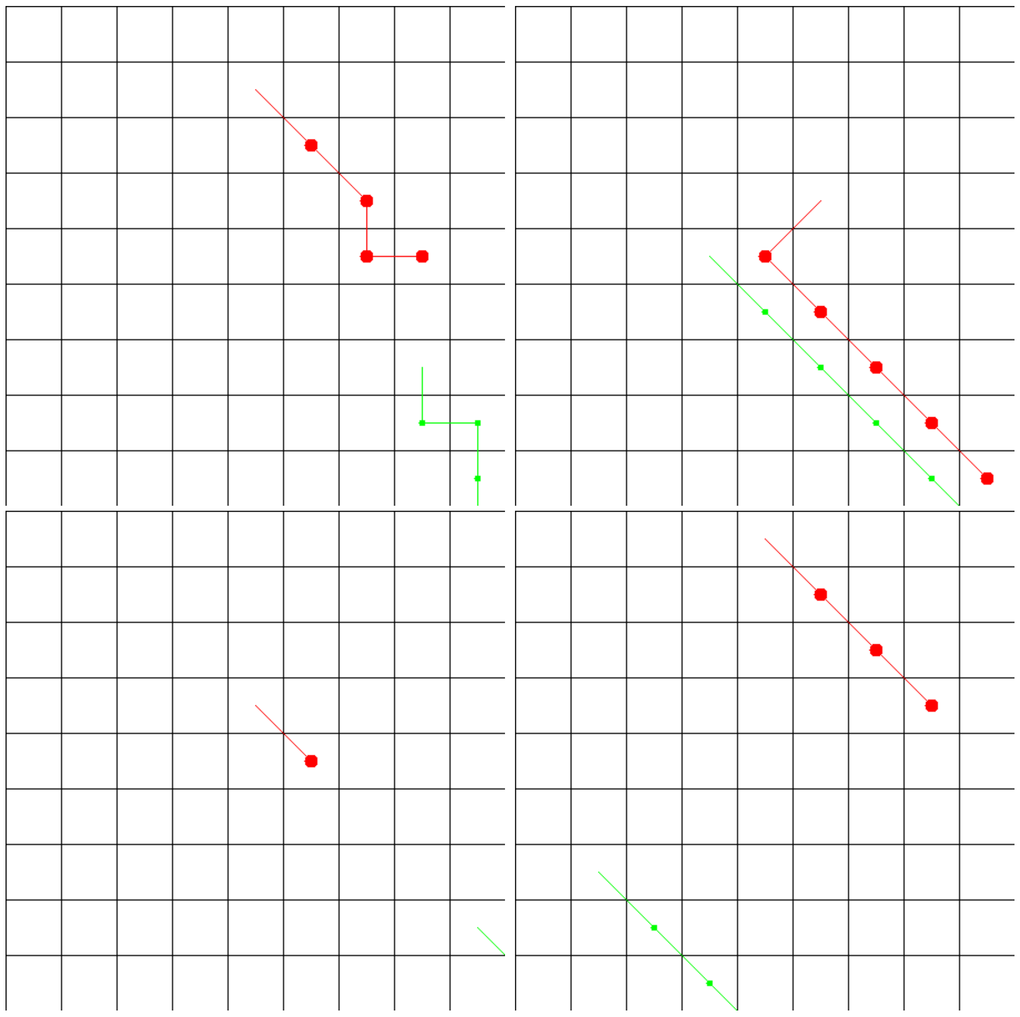
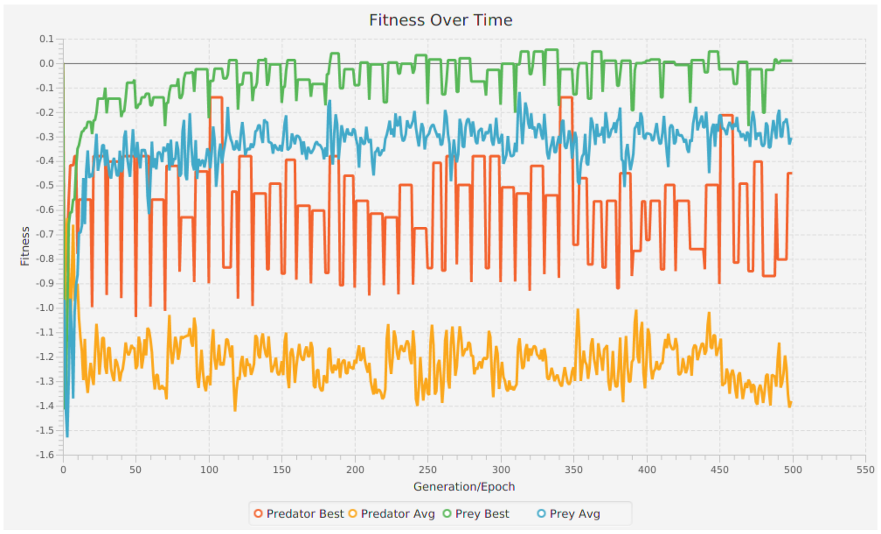
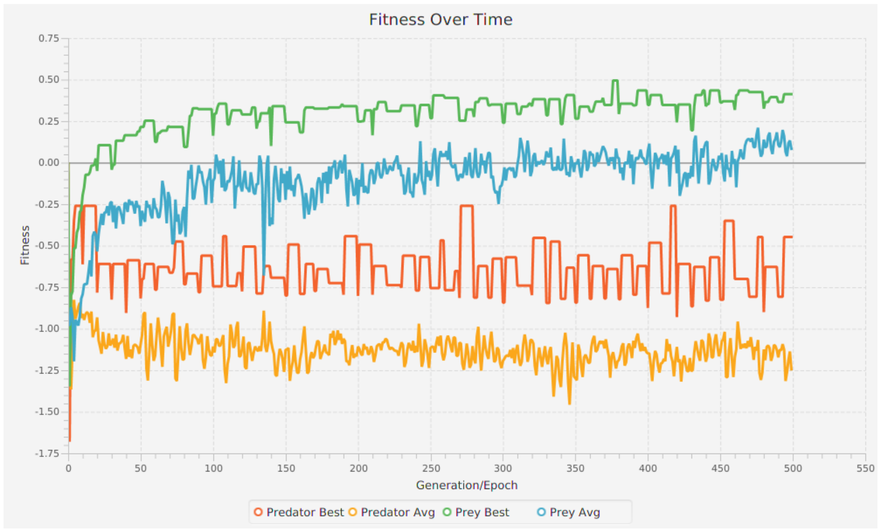


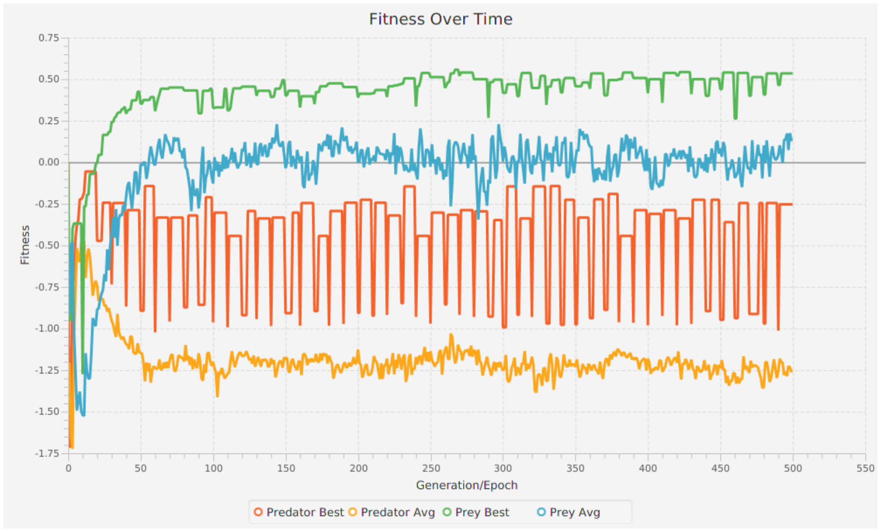
Figure 15: Game V1 Experiment 1: Examples of prey games



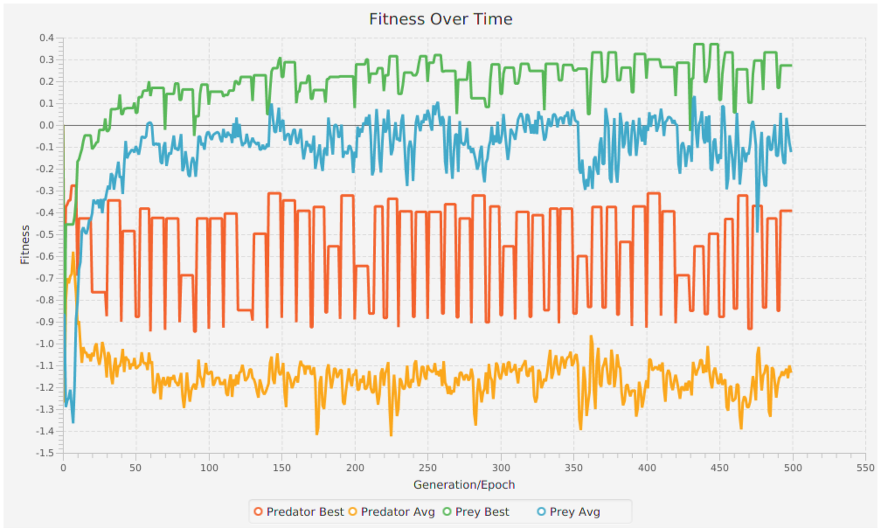
(a) V2 experiment 6 run 3



(b) V2 experiment 8 run 5



(a) V2 experiment 15 run 1



(b) V2 experiment 15 run 4

Figure 17: Examples of Game V2 fitness over time

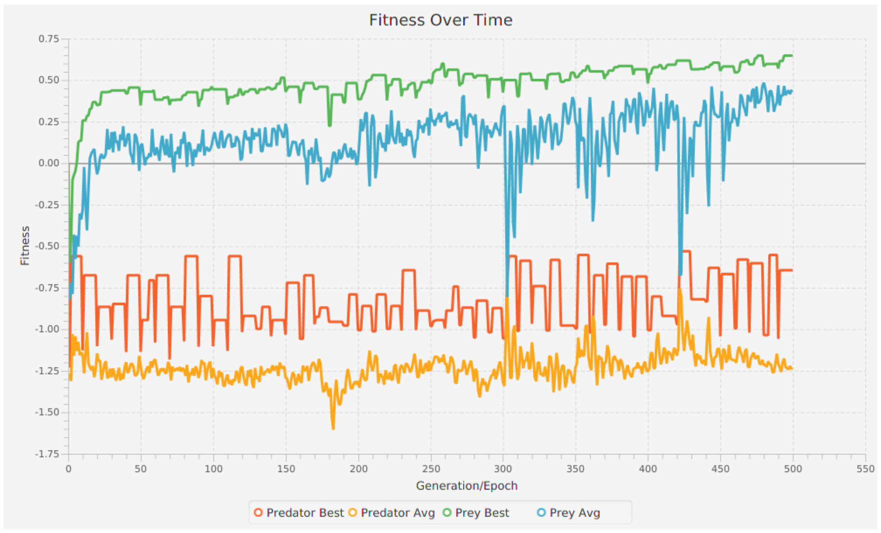


Figure 18: Game V2 experiment 7 run 3: Example of large fitness drop in prey

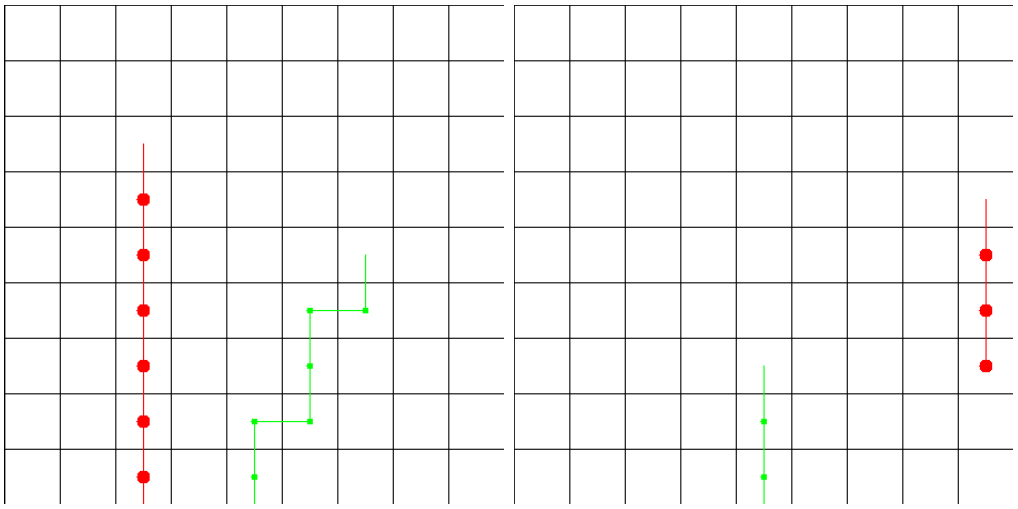


Figure 19: Game V2 experiment 15 Run 1: Example Games

Observing the games played by the predators in experiment 15 run 1, the predator with one of the highest overall fitnesses can be observed. The games show that the reason this particle received a higher fitness value is because it mostly remained on the board with a couple of captures. As experiment 7 run 3 had the highest fitness value right at the beginning of the run, the movements made by the predator and prey are chaotic with no real structure with the results being similar to game V1 where both the predator and prey just pick a direction to move in and they move in that direction until the game ends (see Figure 19).

Taking a look at Game V2 experiment 6 run 3 (as seen in Figure 16b) it can be seen that

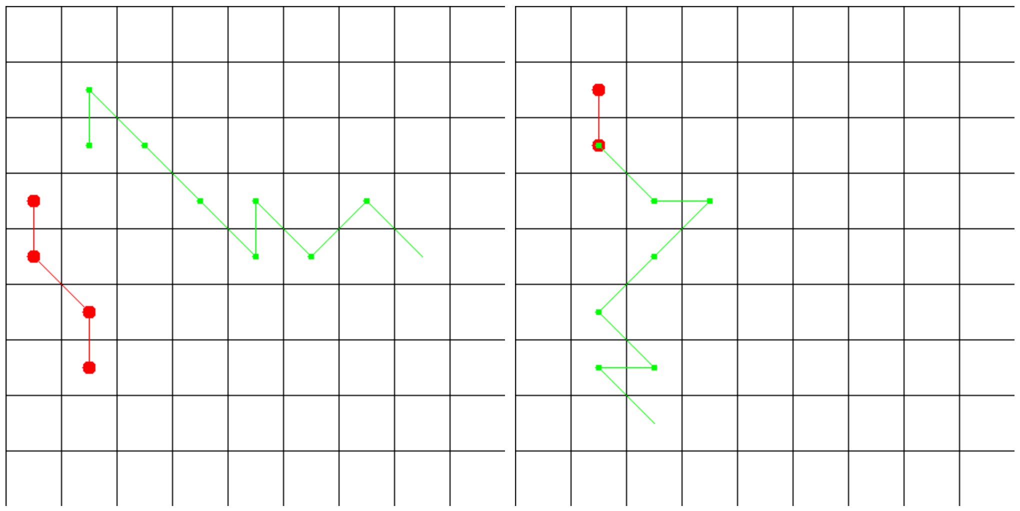


Figure 20: Game V2 experiment 6 run 3: Predator Patrol Example Games

the predator with the overall highest fitness appeared later on in the run. This predator adopted a patrolling approach. In most of the runs, the predator tended to move back and forth on the same line and hope that the prey would come that way (see Figure 20). In most cases this did not work and the prey just ran off the board. The major development for the predator is that in most cases, the predator remained on the board.

As it can be seen in all of the graphs for V2, the prey score has lowered. It is no longer close to 1 as the prey is now penalized for falling off the board. One of the best tactics that the prey evolved is to match the predator's patrol tactic. Using this tactic, the prey just wait for the game to end by patrolling their own area as the predator will never approach the prey anyways. This tactic in can be seen in Figure 21.

The summary results for game V2 can be seen in Table 8. The results are much more varied in this version compared to the results seen in game V1, however in game V2, both the predator and prey achieved better scores on average using the core radius of 5 and perception limit of 30 parameter set. This seems to suggest that allowing the particles to converge closer together before repelling is more beneficial than expanding the search space.

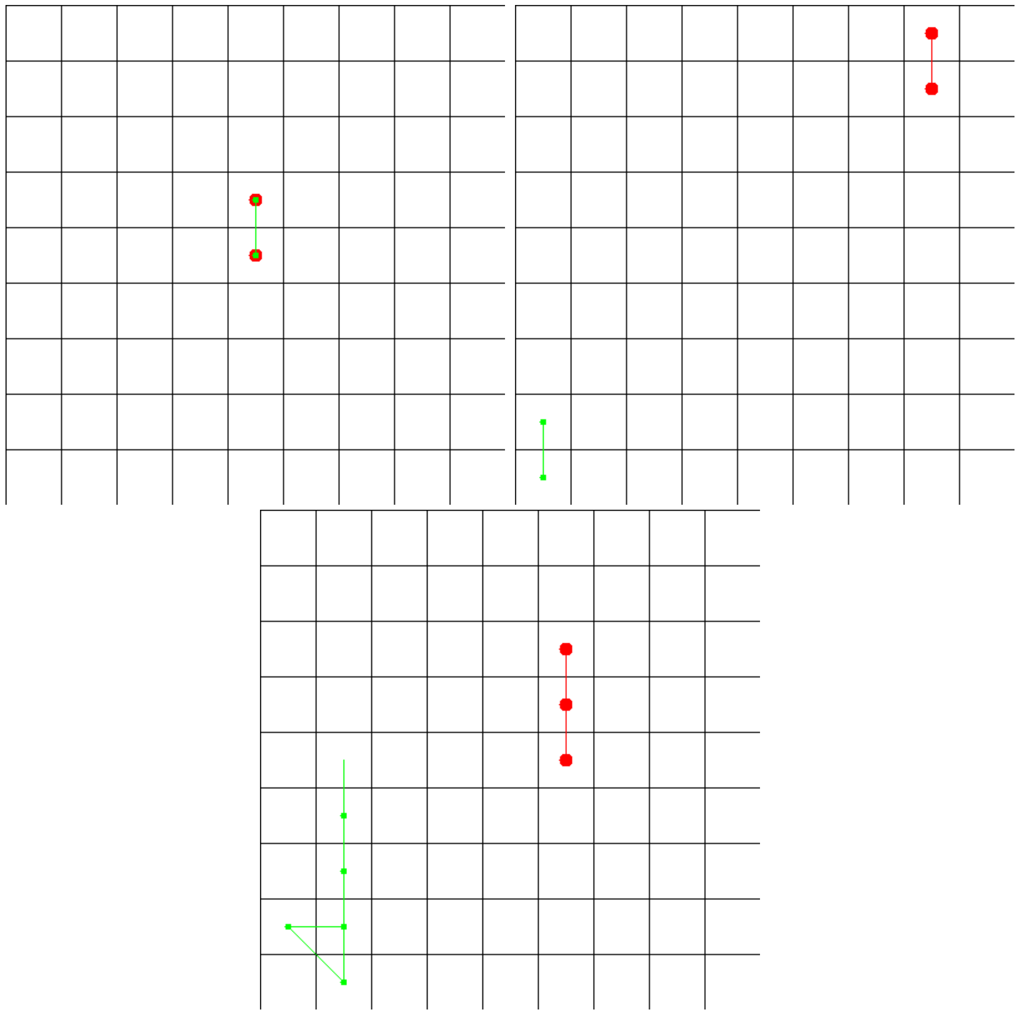


Figure 21: Game V2 Experiment 16 run 5: Prey Patrol Example Games

Experiment	Predator			Prey		
	Max	Average	Min	Max	Average	Min
1	-0.300	-0.384	-0.488	0.576	0.373	0.237
2	-0.308	-0.374	-0.500	0.346	0.259	0.156
3	-0.350	-0.450	-0.568	0.428	0.289	0.183
4	-0.315	-0.392	-0.515	0.471	0.252	0.044
5	-0.285	-0.420	-0.633	0.576	0.425	0.165
6	-0.140	-0.276	-0.460	0.385	0.192	-0.118
7	-0.055	-0.409	-0.563	0.507	0.319	0.122
8	-0.260	-0.393	-0.490	0.418	0.322	0.210
9	-0.175	-0.363	-0.570	0.540	0.381	0.316
10	-0.160	-0.322	-0.468	0.519	0.407	0.298
11	-0.275	-0.413	-0.598	0.497	0.318	0.106
12	-0.330	-0.421	-0.560	0.505	0.294	0.179
13	-0.130	-0.373	-0.713	0.531	0.378	0.148
14	-0.345	-0.459	-0.653	0.566	0.432	0.298
15	-0.278	-0.414	-0.598	0.455	0.288	0.065
16	-0.305	-0.443	-0.520	0.510	0.410	0.315
17	-0.365	-0.489	-0.543	0.520	0.439	0.180
18	-0.345	-0.405	-0.450	0.485	0.354	0.281
19	-0.173	-0.361	-0.533	0.486	0.264	-0.020
20	-0.220	-0.388	-0.490	0.479	0.363	0.204
21	-0.268	-0.449	-0.660	0.619	0.459	0.265
22	-0.280	-0.413	-0.738	0.446	0.293	0.144
23	-0.255	-0.356	-0.443	0.435	0.368	0.260
24	-0.215	-0.270	-0.345	0.480	0.358	0.165

Table 8: Game V2 Summary

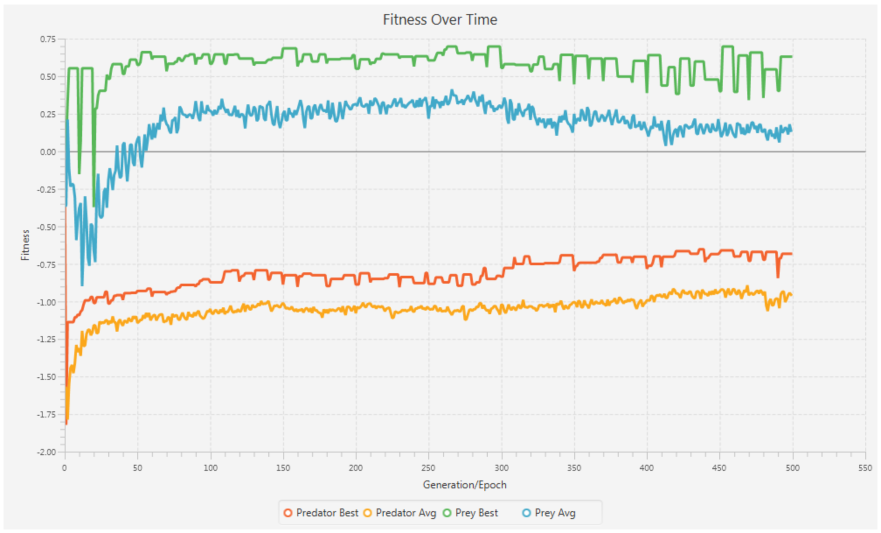
5.4 Game V3

The game rules of Game V3 provide a slight advantage to the prey as if the prey run off of the board, then the predator's score decreases as they could not catch the prey. However, this rule should prevent the patrolling behaviour that was developed in Game V2. Using these rules, most of the predator experiments had trouble reaching a fitness value of -1, while only a couple of runs reached a fitness value of -0.75 (see Figures 22 and 23). The prey on the other hand improved significantly from game V2. The prey fitness with these game rules hovers close to 1 with the fitness only going below 0.75 on rare occasion.

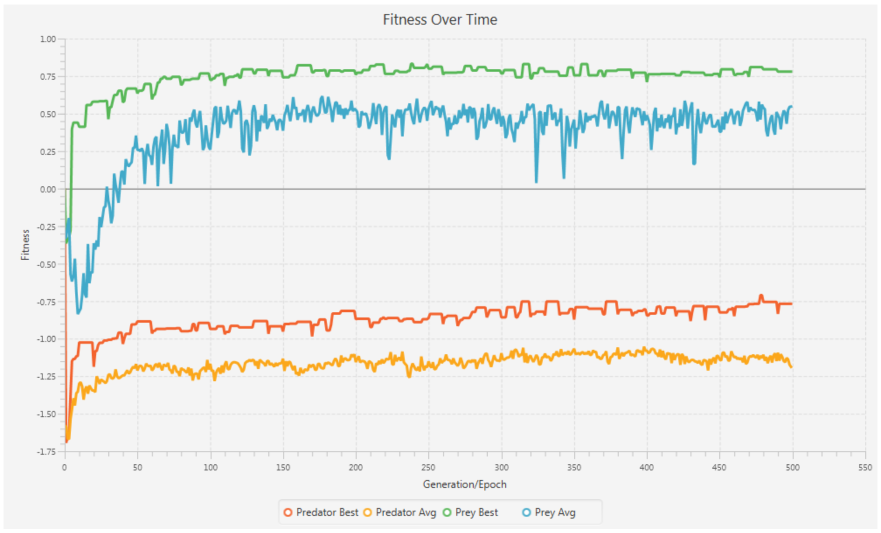
As it can be seen, experiment 23 seems to provide the best results for this set of game rules for the predator as there are two instances where the best fitness surpasses -0.75. Some games played with complex behaviour in experiment 23 run 4 can be seen in Figure 24. While the predator is not the best at catching the prey, in some games, complex patterns start to emerge. Unfortunately and interestingly most games end with the prey leaving the board (examples in Figure 25). This could be due to the fact that the prey is exploiting the rules of game V3. By leaving the board, the prey can be at a net difference in fitness of -1, however if the predator succeeds in catching the prey, then the prey would be at a net different of -2, therefore it is more beneficial for the prey if it runs off the board. Therefore, the change in game rules that was added to try to motivate the predator to capture the prey was in turn used against the predator. It can also be seen that even when the prey is in the process of running off the board, the predator is approaching the prey, which is consistent with the walled game.

It should be noted that in all graphs in Figure 22, the prey average converges slower to the prey best. This seems to suggest that having most of the population be poor in the beginning provides an advantage to the prey to prevent the predators from learning. Comparing the way the prey converges to the graphs in Figure 16b this is even more evident.

In Table 9, the prey tends to have higher average results when comparing 0% charged swarms to 100% charged swarms. This is because the prey's fitness is much better than

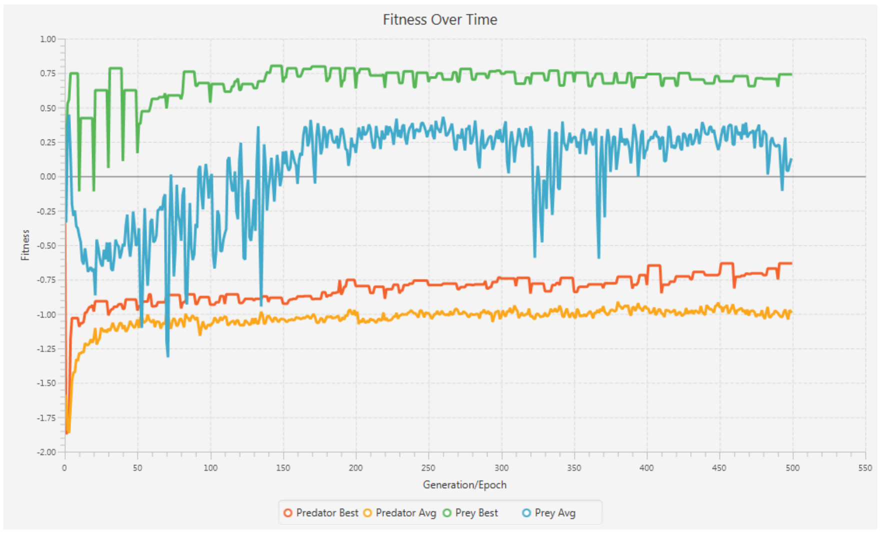


(a) Experiment 2 Run 5

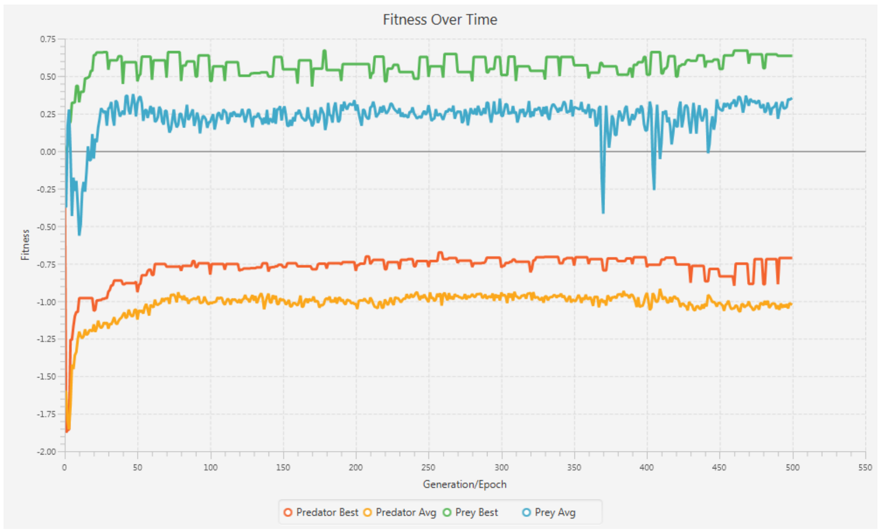


(b) Experiment 3 Run 4

Figure 22: Game V3 fitness over time examples above -0.75

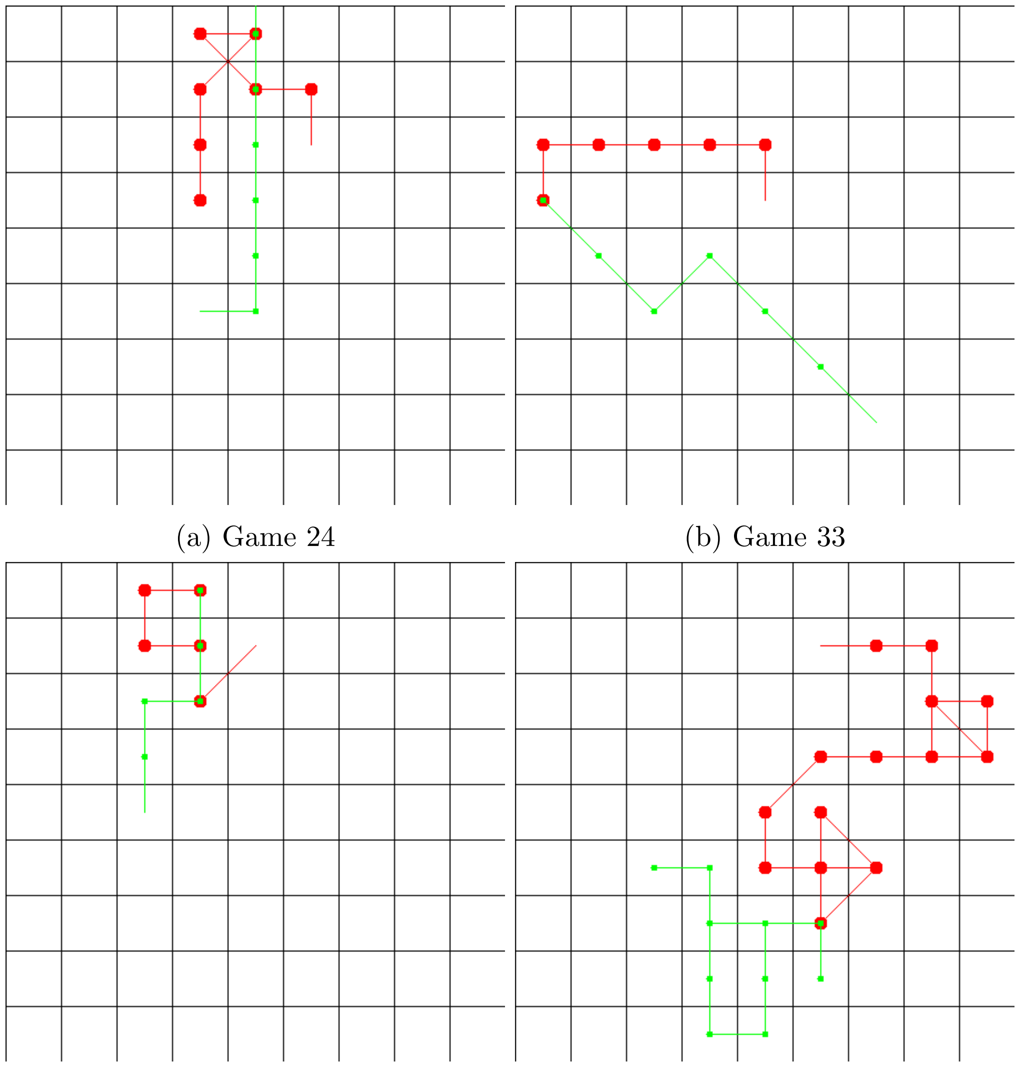


(a) Experiment 23 Run 3



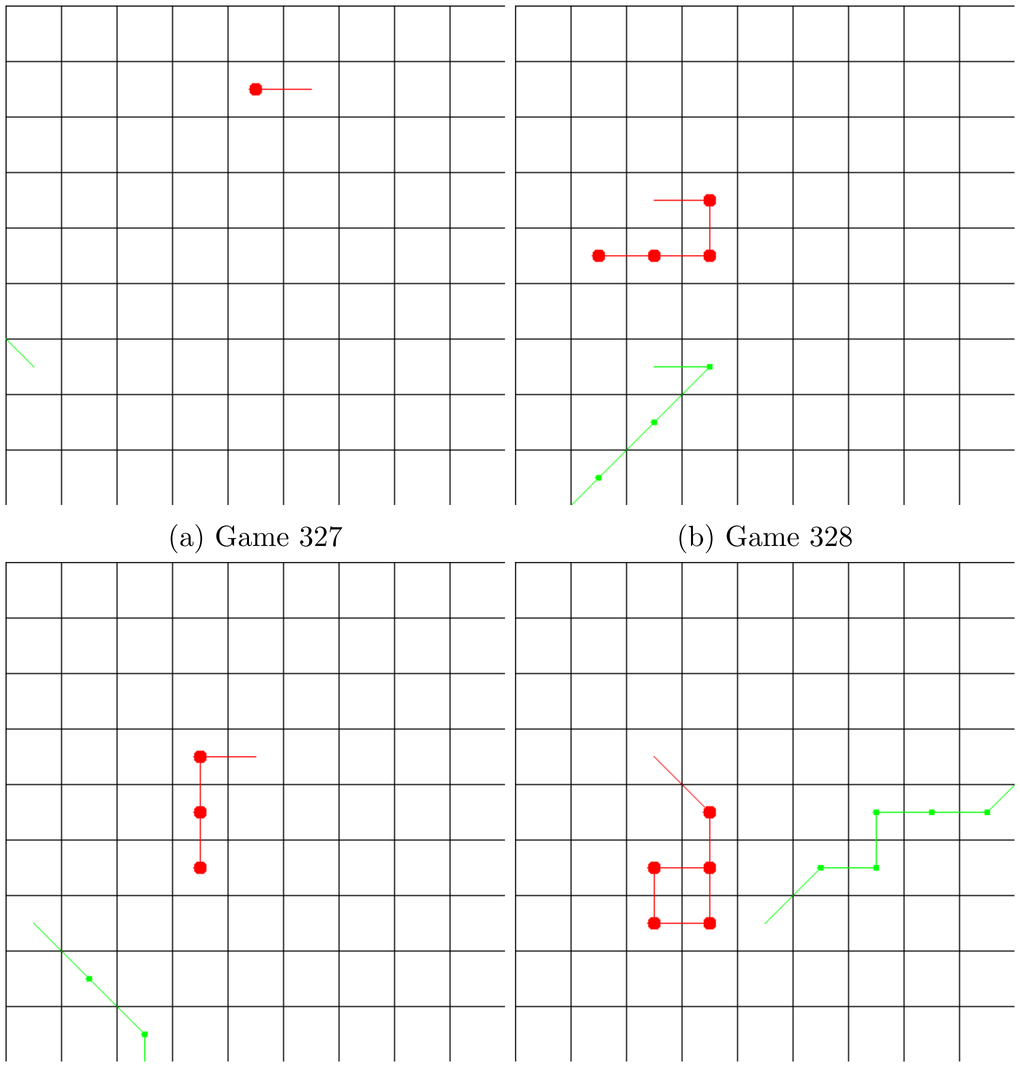
(b) Experiment 23 Run 5

Figure 23: Game V3 fitness over time examples above -0.75 continued


(a) Game 24 (b) Game 33

(c) Game 45 (d) Game 91

Figure 24: Game V3 Experiment 23 Run 5: Predator Game Examples


(a) Game 327 (b) Game 328

(c) Game 329 (d) Game 330

Figure 25: Game V3 Experiment 23 Run 5: Prey Running off of the Board Examples

Experiment	Predator			Prey		
	Max	Average	Min	Max	Average	Min
1	-0.755	-0.866	-0.923	0.817	0.798	0.745
2	-0.653	-0.850	-0.975	0.885	0.690	0.412
3	-0.710	-0.846	-0.893	0.766	0.612	0.520
4	-0.903	-0.920	-0.940	0.821	0.674	0.580
5	-0.803	-0.894	-0.958	0.898	0.816	0.648
6	-0.788	-0.873	-0.955	0.777	0.699	0.562
7	-0.783	-0.821	-0.900	0.732	0.548	0.395
8	-0.753	-0.889	-0.943	0.844	0.788	0.670
9	-0.833	-0.900	-0.948	0.887	0.775	0.637
10	-0.785	-0.893	-0.940	0.899	0.761	0.547
11	-0.773	-0.848	-0.940	0.859	0.657	0.509
12	-0.858	-0.869	-0.885	0.746	0.640	0.467
13	-0.868	-0.919	-0.968	0.843	0.819	0.735
14	-0.718	-0.879	-0.963	0.868	0.754	0.604
15	-0.850	-0.889	-0.925	0.831	0.766	0.688
16	-0.888	-0.910	-0.933	0.842	0.732	0.573
17	-0.665	-0.816	-0.920	0.897	0.770	0.649
18	-0.903	-0.929	-0.955	0.867	0.779	0.729
19	-0.883	-0.900	-0.928	0.856	0.774	0.649
20	-0.763	-0.862	-0.905	0.824	0.708	0.498
21	-0.735	-0.877	-0.935	0.856	0.762	0.492
22	-0.835	-0.892	-0.953	0.884	0.722	0.595
23	-0.633	-0.782	-0.900	0.787	0.584	0.380
24	-0.818	-0.925	-0.993	0.852	0.710	0.417

Table 9: Game V3 Summary

the predators, so there is no need to explore more of the error landscape. Interestingly, the predator also follows this pattern even though exploring the error landscape should lead to better solutions.

6 Conclusion

Throughout this research, various parameter sets were tested with several variations of game rules for the predator vs prey game. It was found that the most balanced game rules occurred when the ability to fall off the map did not exist, as this forced the two types of agents to interact. Although the agents were forced to interact, due to the probability of the predator capturing the prey being low, the prey performed better while needing less development. As such the prey behaviour did not evolve to be as complex as the predator behaviour. The prey tended to get stuck in a corner and get caught by the predator.

Game V1 was then introduced as a version of the game where there are no walls blocking the prey. With the walls not blocking the prey from leaving the board, the prey continues similarly to the walled version of the game, except the prey continues to walk off, and since the predator could not catch the prey, the game ends with the prey being the victor. This caused the prey to continuously run off the board as that is the easiest way to win the game for the prey.

In order to stop such a scenario, Game V2 implemented a penalty if an agent falls off the board. This initially provided the predators a better fitness, but as the evolution took place, the prey always ended up out evolving the predators. In this game version, both sides evolved a patrolling technique. This caused most games to end after 20 steps with the prey emerging victorious.

In order to encourage the predator to catch the prey, a penalty was introduced to the predator if the prey ran off the edge in game V3. This was done in hopes that the predator would rush to capture the prey before the prey fell. However the prey ended up exploiting this new game rule. In most of the experiments, the prey score was high, and when looking at the games, the prey tends to run off the edge of the board. This stifles the learning of the predator enough that the prey end up outsmarting the predators.

7 Future Work

In terms for future expansions, the most critical would be testing an expanded neural network in order to see if the behaviour becomes more complex. The next step after that would be introduce a line of sight in order to restrict both agents to incomplete information of the game state, along with introducing multiple agents from both types into the simulation to observe if any teamwork behaviours evolve.

Further expansions to the system could include removing the chase ability of the predator in order to reduce the network complexity. A border around the board to indicate the edge of the board, or terrain modifications could be added. In addition, multiple board sizes could also be tested. Additional game rule variations should also be considered.

References

[1] F. van den Bergh, A.P. Engelbrecht, ”A study of particle swarm optimization particle trajectories”, `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.142.2258&rep=rep1&type=pdf`, [Online, accessed 2-May-2017].

[2] LH. Langenhoven, ”Evolving Behaviour through Competition Special Project”, Master’s thesis, University of Pretoria, November 2006.

[3] C. Scheepers, ”Coevolution of Neuro-controllers to Train Multi-Agent Teams from Zero Knowledge”, Master’s thesis, University of Pretoria, July 2013.

[4] Apache Commons: Commons Math, `http://commons.apache.org/proper/commons-math/userguide/linear.html`, [Online, Accessed 4-May-2017].

[5] S.B. Maind, P. Wankar, ”Research Paper on Basic of Artificial Neural Network”, in *International Journal on Recent and Innovation Trends in Computing and Communication*, Vol. 2, Issue. 1, pp. 96-100, 2014.

[6] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016. `http://www.deeplearningbook.org`, [Online, Accessed 4-May-2017].

[7] J. Brownlee, ”Particle Swarm Optimization Pseudocode”, `http://www.cleveralgorithms.com/nature-inspired/swarm/pso.html` [Online, Accessed 2-May-2017].

[8] G. Orr, ”A Simple Artificial Neuron”, `https://www.willamette.edu/~gorr/classes/cs449/ann-overview.html` [Online, Accessed 2-May-2017].

[9] S.H. Chun, ”Coursera Neural Networks for Machine Learning Week1 - Neural Network and Machine Learning”, `http://sanghyukchun.github.io/39/` [Online, Accessed 2-May-2017].

[10] J. Kennedy, R. Eberhart, "Particle Swarm Optimization", 1995, `https://www.cs.tufts.edu/comp/150GA/homeworks/hw3/_reading6\%201995\%20particle\%20swarming.pdf`, [Online, Accessed 4-May-2017].

[11] A.P. Engelbrecht, "Particle Swarm Optimization in Dynamic Environments", `http://goanna.cs.rmit.edu.au/~xiaodong/aciss09/tute-slides/dynamicPSO.pdf` [Online, Accessed 2-May-2017].

[12] C. D. Rosin, R. K. Belew, "New Methods for Competitive Coevolution", Unknown, `http://www.sci.brooklyn.cuny.edu/~sklar/teaching/f05/alife/papers/rosin-96coev.pdf`, [Online, Accessed 4-May-2017].

[13] O. E. David et al., "Genetic Algorithms for Evolving Computer Chess Programs", in *IEEE Transactions on Evolutionary Computation*, Vol. 18, Issue. 15, pp. 779 - 789, 2014.

[14] A. Lubberts, R. Miikkulainen, "Co-Evolving a Go-Playing Neural Network", In *Co-evolution: Turning Adaptive Algorithms upon Themselves, Birds-of-a-Feather Workshop, Genetic and Evolutionary Computation Conference (Gecco-2001)*, 2011, `www.cs.utexas.edu/~ai-lab/pubs/lubberts.coevolution-gecco01.ps.gz`, [Online, Accessed 5-May-2017].

[15] C. H. Yong, R. Miikkulainen, "Cooperative Coevolution of Multi-Agent Systems", Unknown, `http://nn.cs.utexas.edu/downloads/papers/yong.tr287.pdf`, [Online, Accessed 5-May-2017].

[16] A. Runka, "Genetic Programming for the RoboCup Rescue Simulation System", Master's Thesis, 2010, `http://rd.library.brocku.ca/bitstream/handle/10464/3184/Brock_Runka_Andrew_2010.pdf?sequence=1`, [Online Accessed 5-May-2017]

[17] G. I. Hawe et al., "Agent-based simulation of emergency response to plan the allocation of resources for a hypothetical two-site major incident" in *Engineering Applications of*

Artificial Intelligence, Vol. 46, Part. B, pp. 336–345, 2015, http://www.sciencedirect.com/science/article/pii/S0952197615001451, [Online, Accessed 5-May-2017]

[18] K. O. Stanley, R. Miikkulainen, "Evolving Neural Networks through Augmenting Topologies", 2002, http://nn.cs.utexas.edu/downloads/papers/stanley.ec02.pdf, [Online, Accessed 5-May-2017]

[19] P. Koehn, "Combining Genetic Algorithms and Neural Networks: The Encoding Problem", Master's thesis, The University of Tennessee, 1994, http://homepages.inf.ed.ac.uk/pkoehn/publications/gann94.pdf, [Online, Accessed 5-May-2017]

[20] Y. Lecun et all., "Efficient BackProp", Unknown, http://yann.lecun.com/exdb/publis/pdf/lecum-98b.pdf, [Online, Accessed 5-May-2017]

[21] B. A. Garro et al., "Evolving Neural Networks: A Comparison between Differential Evolution and Particle Swarm Optimization", Unknown, https://link.springer.com/chapter/10.1007/978-3-642-21515-5_53, [Online, Accessed 5-May-2017]