The Red Queen's Hypothesis: *Does Selection Pressure Affect Co-evolution in a Predator-Prey Model?*

Vineel Nagisetty
Electrical and Computing Engineering
University of Waterloo
Waterloo, Canada
vineel.nagisetty@uwaterloo.ca

***Abstract*—The Red Queen's Hypothesis (RQH) proposes that in predator-prey relationships, the selection pressure exerted on each species by the other is an important factor causing both species to evolve. Specifically, this selection pressure sets off an 'arms race' where the predators learn to better hunt the prey while the prey learn to better evade predators - leading to co-evolution. However, it is not clear as to exactly how this selection pressure affects co-evolution. Specifically, how does the strength of the selection pressure exerted by the predator on the prey affect the overall co-evolution behaviour? This work aims to answer this question, simulating a simplified predator-prey model using open AI gym for simulation and NeuroEvolution of Augmenting Topology (NEAT) for evolving the predator-prey models. Results indicate that lower selection pressure allows for a more stable co-evolution with higher fitness scores for the best individuals while higher selection pressure results in more diverse behaviours. This suggests that tuning selection pressure may allow for better focused training of predator-prey models - especially for robotic applications.**

***Index Terms*—predator-prey model, artificial life, co-evolution**

I. Introduction

The Red Queen's Hypothesis (RQH), introduced by Van Valen, proposes that "organisms must constantly adapt, evolve and proliferate in order to survive while pitted against ever-evolving opposing organisms in a constantly changing environment, as well as to gain reproductive advantage" [1]. The selection pressure exerted on an organism by an 'ever-evolving' opposing organism is an important factor leading to its evolution. Nowhere is this phenomena more apparent than in predator-prey relationships, where the interactions between the predator and prey species affect the fitness of both species (here fitness refers to the population size of each species). There is a perceived 'arms race' between the two species where the predators seek to improve their behaviour and capability to hunt prey while the latter aim to better evade the former. There has been extensive research supporting RQH in the context of predator-prey interactions in nature, from some of the smallest life forms (bacteria [2]) to some of the largest ones (whales [3]) - and many more in between [4]–[7].

However, it is not clear as to what extent the strength of selection pressure found in predator-prey relationships affects their co-evolution. In other words, for a given prey, does a 'stronger' predator (here a stronger predator refers to a more evolved one and is considered to exert a greater selection pressure on the prey) cause the prey to evolve any differently compared to a 'weaker' predator? The research questions addressed in this paper are the following: does the strength of selection pressure exerted by the predator on the prey in a predator-prey relationship affect their co-evolution? If so, how is this co-evolution different and can we leverage insights gained by answering this to other settings?

This project seeks to answer the aforementioned question using Artificial Life (AL) models. AL models aim to simulate emergence of complex global behaviours using interactions between relatively simpler components or agents [8], [9]. Often, AL experiments seek to create models of minimal complexity that can still result in emergence of the desired complex behaviour. Specifically, predator-prey relationships are simulated using predator-prey models. There are several applications of such predator-prey models - most notably in the field of robotics. In particular, it has been observed that "co-evolved predator robots outperform predators evolved against fixed prey." [10]. This shows the advantage of simulating co-evolution to train predator-prey models. Optimal predator models can chase after targets efficiently, making them suitable for robots that are required to reach specific goal posts or chase after specific targets. On the other hand, optimal prey models are great at avoidance, making them ideal for robots such as unmanned underwater vehicles (uuv) that need to escape from underwater predators.

The goal for this project is to create a simplified simulation of a predator-prey model where the predator species exhibit varying degree of selection pressure and analyze their co-evolution results. Both the predator and prey species are evolved using NEAT [11], an algorithm that allows for evolving neural networks with increasing complexity - which are used to control behaviour of predators and preys. The project simulated the predator-prey model using a third party environment for Open AI Gym [12].

A. Contributions:

1) The key contribution for the project is to show that varying the degree of selection pressure exerted by the predator on the prey results in a difference in their co-evolution. Specifically, the experiments demonstrate that lesser selection pressure results in less variance in co-evolution with higher fitness values for the best individuals while higher selection pressure results in more diverse behaviours (see Section IV-C).
2) The environment used for simulation was modified in order to use NEAT (see Section III-C).
3) An extensive search was conducted to find parameters that result in successful co-evolution in this setting. (see Section IV).

II. Related Work

Modelling co-evolution in predator-prey models is an active area of research. A significant amount of literature involves simulating predator-prey interactions using numerical equations [13]–[16]. In this setting, the behaviour of predator and prey species are often defined via a system of equations and, after applying the equations a specific number of iterations, the resulting population size for each species is considered.

Similarly, there have been other approaches in simulating predator-prey models. Initial research simulating predator-prey models via evolving agents started in the early 1990s [17]–[20] and were later extended [21]–[23]. A majority of these approaches use evolutionary and genetic algorithms to co-evolve predator and prey species. These simulations were later extended to real robots in [24]–[26]. A current survey of state of the art methods can be found in [27].

NeuroEvolution of Augmenting Topologies (NEAT), an algorithm introduced by Stanley and Miikkulainen, allows for the evolution of neural networks of increasing complexity to simulate complex behaviour that mimics evolution found in nature [11]. It has been used in several settings, such as in evolving complex robot behaviours [28]–[31]. Notably, Baer in [32] used NEAT to simulate co-evolution of multi-agent systems. In this setting, the prey had a fixed policy while several predators, each with a unique controller applied NEAT in order to learn how best to co-operate to catch the prey.

Finally, Chen in [33] utilized NEAT to evolve both predator and prey species - which to the best of my knowledge is the only other research applying NEAT to evolve both predator and prey species simultaneously. While this project utilizes NEAT algorithm to achieve the same affect, it differs in several key aspects from [33]:

1) This project varies the degree of selection pressure to analyze changes in co-evolution, while the work in [33] varies the number of predator agents to observe the change in co-evolution.
2) The environment and software agents used for simulation are different.
3) This work analyzes the variance and diversity of behaviour for predator prey species while [33] does not.

III. Implementation

A. Simulation Environment

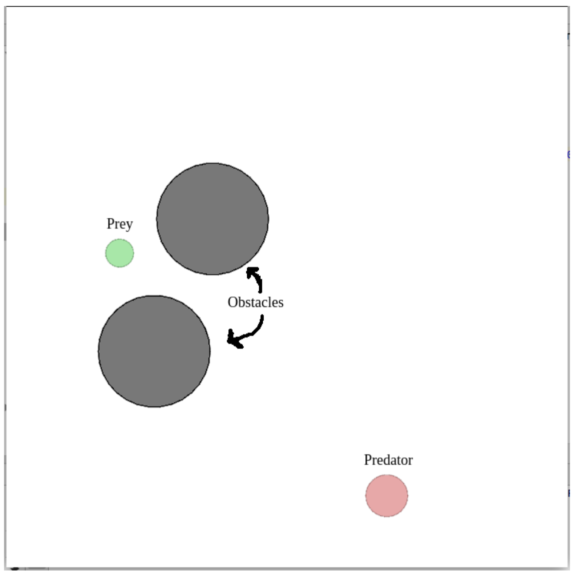

Fig. 1. Example of a figure caption.

This project utilizes a third party Open AI Gym environment [12] that was used for other works such as [34]–[36]. Specifically, a new map was created, inspired by code from [12], however, a non trivial part was modified such as customizing the map as well as the fitness functions to run this project. The characteristics of the environment are as follows:

1) **Map:** the map (shown in Figure 1) is a 700 x 700 square that contains a predator agent (shown in red), a prey agent (shown in green) and two obstacles (shown in dark grey). At the start of a simulation, the predator, prey and obstacles are randomly placed on the map. Note that there are no boundaries for the map, so predators and prey can leave the map. To discourage this action, the fitness function for each species assigns a negative reward for leaving the map.
2) **Sensors:** both the predator and prey are equipped with sensors that are equidistantly located throughout their body. The predator is equipped with twelve while the prey has ten sensors - to account for their relative size difference. The environment is partially observable, meaning these sensors indicate the distance from the particular agent to the nearest object (which can be an obstacle or the other agent). They have a maximum range of 100.
3) **Motors:** both the predator and prey have a motor that takes as input a two dimensional vector with values in range of $[-1, 1]$ corresponding to the desired velocity in the x and y axis. This velocity value is added to the agent's existing velocity, with a limit to the maximum acceleration and velocity, ensuring that both predator and prey have the same maximum speed. A negative velocity value in the x axis causes the agent to move

Algorithm 1: Predator fitness function

```
input  : Predator location $T\_loc$
input  : Prey location $Y\_loc$
output: Reward $r$
1 if $T\_loc$ is out of the map then
2     $r = -1*$ distance from map;
3 else if $T\_loc == Y\_loc$ then
4     $r = 10$;
5 else
6     $r = -0.1 * \sqrt{T\_loc^2 - Y\_loc^2}$;
7 end
8 Return $r$
```

Algorithm 2: Prey fitness function

```
input  : Prey location $Y\_loc$
input  : Predator location $T\_loc$
output: Reward $r$
1 if $Y\_loc$ is out of the map then
2     $r = -1*$ distance from map;
3 else if $T\_loc == Y\_loc$ then
4     $r = -10$;
5 else
6     $r = 0.1 * \sqrt{T\_loc^2 - Y\_loc^2}$;
7 end
8 Return $r$
```

left, while a positive velocity value causes it to move right. Similarly, a negative velocity in the y axis causes the agent to move down while a positive value causes it to move up. A value of 0 results in the agent to not move along that axis.

4) **Reward:** at a high level, the predator gets a higher reward for colliding with the prey, while getting a lower reward the farther apart it is from the prey. On the other hand, the prey gets a higher reward the farther it is from the predator, while getting a negative reward if the predator collides with it. Both the predator and the prey get a negative reward for leaving the map, discouraging this behaviour. The fitness functions for the predator and prey are given by Algorithms 1 and 2 respectively.
5) **Controller:** the behaviour of each agent is realized via their respective controller. The controller takes as input the sensor information of the agent and produces the values for the motor as output. These controllers are evolved via NEAT to achieve desired behaviour.

At the beginning of each simulation, the predator, prey and the two obstacles are randomly placed on the map. Each simulation runs for a fixed amount of time steps (empirically chosen as 1000). As the simulation unfolds, the predator is tasked with chasing the prey while the goal of the prey is to avoid the predator - akin to a game of tag. Furthermore, both the predator and prey need to avoid colliding with obstacles as well as stay on the map to be efficient in their roles. This allows for the emergence of an interesting co-evolution behaviour where both the predator and prey learn to stay in bounds while they get better at their respective tasks.

Algorithm 3: Single Run of NEAT

```
input  : Population size $N$
input  : Number of Generations $G$
output: Best individual $n^*$
1 Initialize population of neural networks of size $N$
2 foreach generation $g$ in range $G$ do
3     Calculate fitness of all individuals in $N$
4     Perform crossover and mutation
5     Select individuals for next generation
6 end
7 Return best individual $n*$
```

B. NeuroEvolution of Augmenting Topologies

NeuroEvolution of Augmenting Topologies (NEAT), introduced by Stanley and Miikkulainen, allows for the evolution of neural networks by starting with small, simple networks and allowing them to become increasingly complex over training cycles [11]. This parallels how organisms in nature evolve via increasing complexity. At a high level, NEAT uses genetic algorithms for evolving artificial neural networks to achieve increased complexity. This increased complexity is achieved via increasing nodes, connections and biases in the neural networks. It has been used in various research [28]–[32], [37].

Algorithm 3 gives a high level overview of NEAT. The algorithm takes as input the size of the population to generate N, the number of generations to run G and returns the best individual n^* as output. Initially, a population of size N is randomly generated (line 1). This is then used to run G number of iterations (line 2). At each iteration, the fitness for all individuals in N is computed (using Algorithms 1 or 2 for predator or prey respectively). After that, the top individuals are selected, while the rest undergo change in the form of crossover and mutation that updates their network architecture and/or adjusts the weights and biases (line 4). Finally, the individuals for the next generation are selected based on a combination of randomness and their fitness (line 5). A thorough explanation of the crossover, mutation and selection of individuals is out of the scope of this paper and the reader is encouraged to refer to [11].

Note that Algorithm 3 is applied alternatively to the predator and prey so that only one of them is evolving at once, consistent with [32], [33]. This project uses the `neat-python` module that is available as open source for Python [38].

C. List of Implementations

In order to run the experiments, several key parts of the code (found under the specific file in Appendix) were specifically implemented using Python 3.6, such as:

1) The main code that implements the project is found in `project_NEAT.py`

2) The actions generated by the agent were implemented in `utils.py`.
3) Several functions for producing graphs has been added to existing code in `visualize.py`.
4) The map, inspired by code from [12] was modified to suit the project by updating initialization and is found in `scenario/environment.py`.
5) The fitness functions were modified to account for the agent leaving the map as well as updating based on the distance to the adversary and can be found in `scenario/environment.py`
6) Several files had minor changes to work with the current version of Python (in files such as `render.py`).

IV. Experimental Results

Initially, a considerable amount of effort was put in to find parameters that result in an emergence of co-evolution, since otherwise the experiment would be meaningless. Even though the simulation is simplified, this task still has a high degree of complexity due to the huge search space resulting from the many number of tunable parameters in NEAT and in the environment. Furthermore, there is a significant chance of failure in adversarial training settings (such as co-evolution). Here, an emergence of co-evolution is verified through a visual inspection of the simulation where the predator was actively chasing the prey while the prey actively avoided the predator while both tried to stay on the map and avoid obstacles. After a non-trivial amount of parameter search, the parameters were set for both the predator and prey. A sample of these parameters can be found in Table I. Note that due to space restrictions, only the most important parameters are given in the Table. Please refer to the files `predatorConfig` and `preyConfig` for the full list of parameters for the predator and prey respectively. It is important to note that these may not be optimal parameters, but they result in emergence of co-evolution behaviour most of the time (=> 80%).

TABLE I
NEAT Parameters for Predator and Prey

Parameter	Value
Population size	30
Probability of Adding Connection	0.15
Probability of Deleting Connection	0.1
Probability of Adding Node	0.15
Probability of Deleting Node	0.1
Bias Replacement Rate	0.02
Bias Mutation Rate	0.8
Weight Mutation Rate	0.8
Weight Replacement Rate	0.02

A. Experiment Setup

As previously stated, the goal of this project is to investigate how the selection pressure exerted by the predator on the prey in a predator-prey relationship affects their co-evolution. Here, selection pressure on the prey refers to how evolved the predator is. In other words, a predator that is more evolved is considered to exert a higher selection pressure on the prey compared to one that is less evolved. Therefore, we measure selection pressure as the number of additional generations the predator has been trained (using NEAT) over the prey. To that end, various number of generations (meaning varying degrees of selection pressure) are selected and the result on the co-evolution behaviour is analyzed (see Section IV-B for more details on the evaluation criteria).

Algorithm 4: Overall algorithm of the experiment

```
input : Number of Generations $G$
input : Number of Trials $T$
input : Population Size $N$
output: Best individual predator $t^*$
output: Best individual prey $y^*$
1 Set number of epochs $E = \frac{200}{G}$;
2 foreach Trial $t$ in range $T$ do
3     foreach Epoch $e$ in range $E$ do
4         $t^*$ := Run NEAT for predators, NEAT(N, G);
5         $y^*$ := Run NEAT for prey, NEAT(N, G) ;
6     end
7 end
8 Return $t^*$ and $y^*$;
```

However, for fairness, the experiments ensure the total number of training generations are equal in all experiment parameters by dividing the total number of training cycles (empirically selected as 200) by the number of generations. This value is referred to as an epoch. In other words, an epoch consists of first training the predator and then the prey for the desired number of generations. Furthermore, each grouping of the parameter was run for ten trials to account for noise. The overall experiment is described in Algorithm 4.

The various settings selected for the experiment can be found in Table II. The first parameter set of (10, 1, 200) (trials, generations, epochs) is the case where both the predator and prey are co-evolving simultaneously (or as simultaneously as the simulation permits), consistent with [32], [33] and is considered to be the default choice. This is the case where the predator exerts the least amount of selection pressure on the prey. The fourth parameter set of (10, 200, 1) lies on the other extreme where the predator is first fully evolved before the prey starts evolving. This is the setting in which the predator exerts the most selection pressure on the prey. The final two parameter sets of (10, 10, 20) and (10, 20, 10) serve as a good contrast since the selection pressure of the predator is almost in the middle of the two extreme cases.

TABLE II
Parameter set chosen for the project.

Trials T	Generations G	Epochs $E = \frac{200}{G}$
10	1	200
10	10	20
10	20	10
10	200	1

B. Evaluation Criteria

Note that the rewards for predator and prey have an inverse relationship. This means that as long as both the predator and prey remain on the map, their average fitness will add up to 0. Since the rewards are inverse, observing the reward of each individual species is not a good indicator of the overall emergence of co-evolution. Instead, combining the average rewards of the predator and prey gives a better indication of co-evolution since if the combination gets close to 0, that means both the predator and prey have learnt to stay on the map while chasing/avoiding each other - a good indication of emergence of co-evolution behaviour. Specifically, the following evaluation criteria is used to analyze the results of the experiment:

1) **Mean overall fitness:** this refers to the combined fitness of the predator and prey, averaged over the number of trials. Here, the trend of the mean overall fitness is a good indicator of the emergence of co-evolution behaviour.
2) **Best overall fitness:** this refers to the combined fitness of the best performing predator and prey, averaged over the number of trials.
3) **Variance of overall fitness:** this refers to the standard deviation of the mean overall fitness, averaged over the number of trials. This can indicate stability of co-evolution, since a low variance implies that the co-evolution behaviour is more stable, and vice versa.
4) **Diversity of controllers:** this refers to the overall diversity of neural network controllers generated by the experiment method, averaged over the number of trials. The diversity of neural network controllers is calculated by the NEAT algorithm (refer to [11] for more details).

C. Results

The results of the mean overall fitness, obtained by combining the predator and prey fitness values for each parameter set and averaging fitness values over 10 trials is given in Figure 2. Note that the graphs (from top to bottom) are in the same order as the parameter settings given in Table II. The top graph, representing (generation, epochs) of (1, 200) is observed to increase the slowest out of the four. However, there is less variance in the graph as it is observed to be 'smoother' compared to the rest of the graphs. By contrast, the graph at the bottom (200, 1) is observed to rise quickly but drops significantly after around 100 training cycles and varies for the rest of the cycles. The graphs in the middle, (10, 20) and (20, 10) show similar trends, with the latter exhibiting more variance. Overall, two trends are clear: 1. The lower the selection pressure, the less variance in the overall average fitness, and 2. the higher the selection pressure, the quicker the initial ascent.

The results of the best overall fitness, obtained by combining the best predator and prey fitness values for each parameter set and averaging fitness values over 10 trials is given in Figure 3. The order is unchanged from before, consistent with Table II. Note that the top graph (1, 200) is much higher than the others, and keep increasing with the number of training cycles - indicating that the best individuals are improving faster and with greater consistency than the other graphs. It is once again observed that the lower the selection pressure (i.e as we go down the graphs), the lower the overall best fitness scores and the higher the variance in the scores.

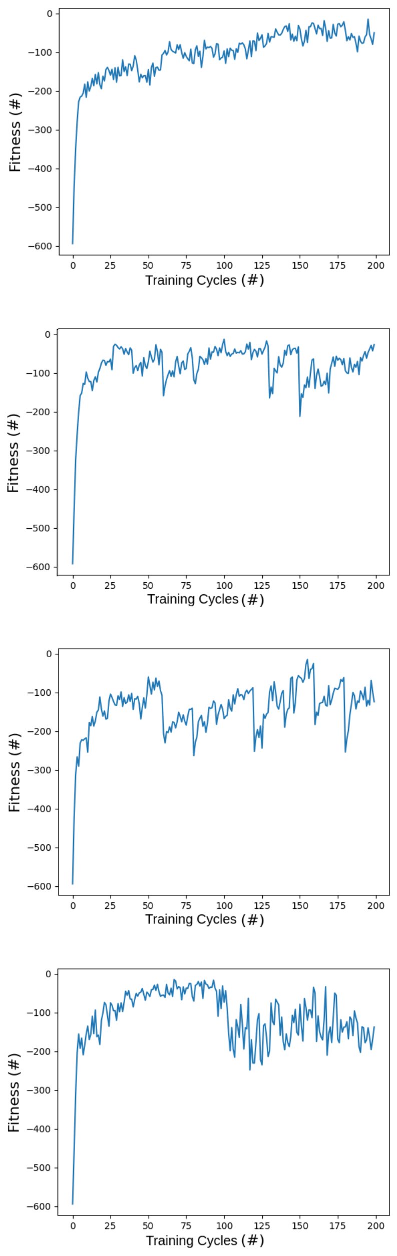

Fig. 2. Results of mean overall fitness. X axis represents the number of overall training cycles while Y axis represents the combined fitness of predator and prey, averaged over 10 trials. From top to bottom, the parameters change as follows (generations, epochs): (1, 200), (10, 20), (20, 10), and (200, 1).

The results of the variance of overall fitness, obtained by combining the standard deviation values for the predator and prey average scores for each parameter set and averaging fitness values over 10 trials is given in Figure 4. The order

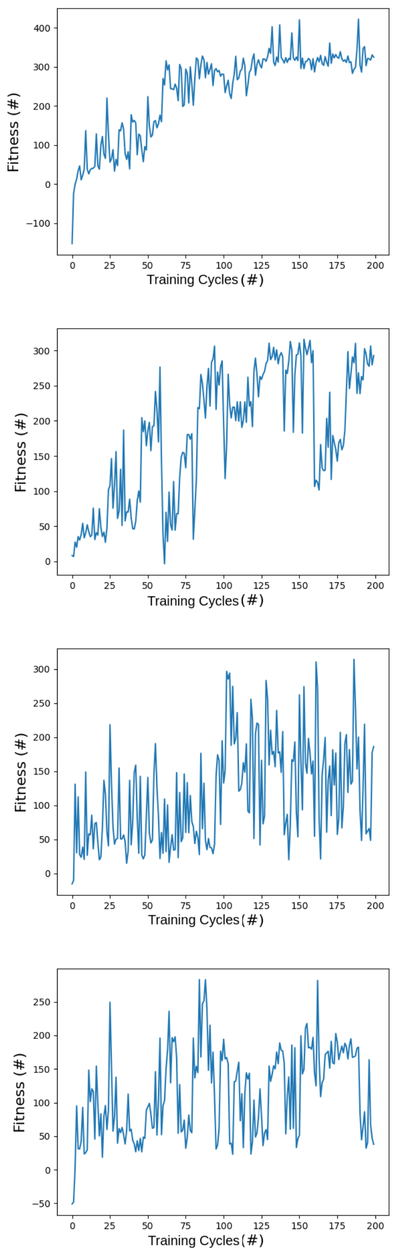

Fig. 3. Results of best overall fitness. X axis represents the number of overall training cycles while Y axis represents the combined fitness of best predator and prey, averaged over 10 trials. From top to bottom, the parameters change as follows (generations, epochs): (1, 200), (10, 20), (20, 10), and (200, 1).

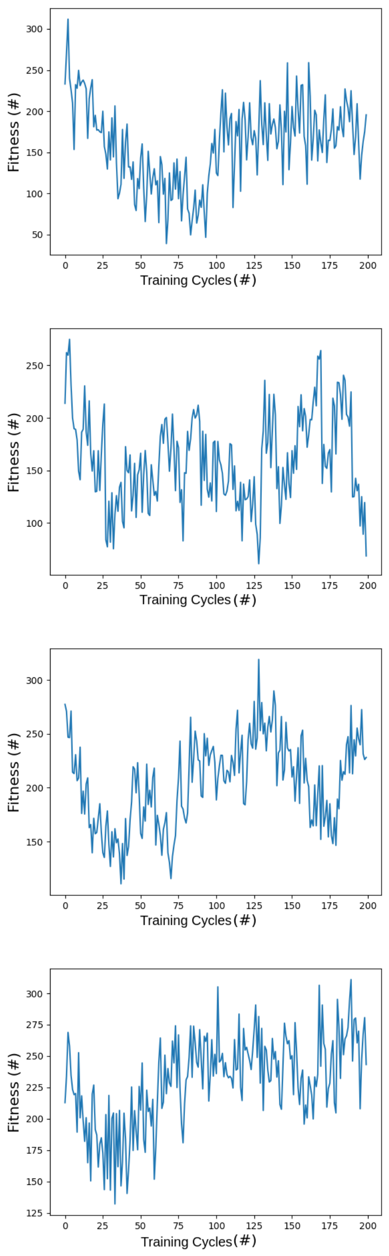

Fig. 4. Results of variance in overall fitness. X axis represents the number of overall training cycles while Y axis the standard deviation of fitness of predator and prey, averaged over 10 trials. From top to bottom, the parameters change as follows (generations, epochs): (1, 200), (10, 20), (20, 10), (200, 1).

of the graphs remains unchanged. Note that the graphs here confirm previous findings: the lower the selection pressure, the lower the variance - especially after 100 training cycles.

What the findings from the previous Figures suggest is that the parameter set of (1, 200) results in the best overall fitness of the parameter sets. Furthermore, there is a clear trend showing that the experiments run with lesser selection pressure result in greater overall fitness. Another observation yields that the lesser the selection pressure, the less the variance in overall fitness - signifying that lesser selection pressure allows for greater overall fitness and more stability in overall fitness.

The results of the diversity of behaviour, obtained by observing the diversity in neural network architectures combining predator and prey species and averaging over 10 trials is given in Figure 5. This diversity is calculated directly by the NEAT algorithm. Note that the number of colors in the graph correspond to the different controllers - so a graph with more colours represents having a greater diversity of controllers. The area of the color represents the population size of each kind of controller. It is apparent that the lower the selection

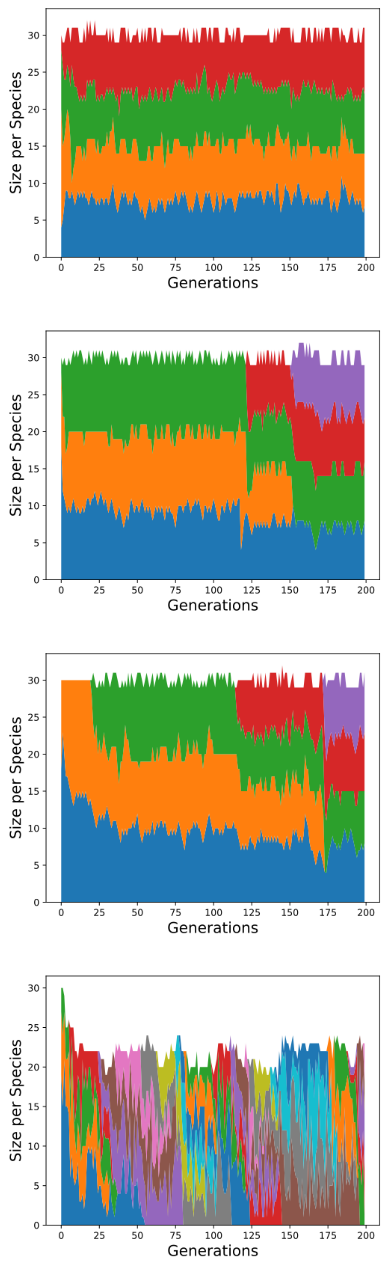

Fig. 5. Results of diversity of neural network controllers for predator prey models. Here the various colors represent different controllers, as calculated by the NEAT algorithm. The width of the color represents the number of population size of individuals with the respective controller. From top to bottom, the parameters change as follows (generations, epochs): (1, 200), (10, 20), (20, 10), and (200, 1).

pressure, the more stable each controllers population size and the lower diversity in controllers. This indicates that higher selection pressure results in more diversity of controllers.

V. Critical Evaluation

Results in IV-C indicate that the strength of selection pressure exerted on the prey by the predator (measured via number of generations of NEAT training) affects their co-evolution behaviour. Smaller number of generations (hence lower selection pressure) are observed to result in more stable fitness with low variance and mostly monotonically increasing fitness values - especially in the case of best fitness. On the other hand, greater number of generations (i.e higher selection pressure) lead to higher variance in learning, where the average fitness seems to initially increase quicker but then drops. This difference in variance may be due to the fact that lower selection pressure on the prey results in a less complex problem. In this setting, small improvements to the prey might make a noticeable difference, resulting in clearer gradients so the evolutionary algorithm can improve easier (via selecting individuals with greater fitness). On the other hand, higher selection pressure results in a much more challenging task and the prey might have to drastically improve to achieve success. In such a setting an individual with a slightly better fitness might fare no better than one with a lower fitness and the species as a whole may need to rely much more on random evolution to improve - resulting in a higher variance.

However, higher selection pressure does seem to have an advantage - particularly in allowing for greater diversity of controllers while maintaining good fitness values. A greater diversity of controllers implies greater diversity of behaviour. There are a class of algorithms known as novelty search which have important applications in robotics since behavioural diversity allows for more robust robots. This may be leveraged in robotic applications where diversity in controller behaviour is often almost as important as fitness.

A visual observation of the simulation using the best predator and prey yielded some interesting results. Overall there were several diverse behaviours observed from both the predator and prey. In one particular setting, the prey seemed to seek obstacles so it could hide from the predator (found in `video_1.mp4`). In another setting, the predator seemed to use the boundary to ‘corner’ the prey (see `video_2.mp4`). Overall this research may be helpful in many robotic applications where chasing or avoiding behaviour is needed. For such requirements, one may train their agent using a predator-prey model, tuning the number of generations and epochs to balance stable training with diverse behaviour.

VI. Conclusion

This project investigated whether the strength of selection pressure exerted by the predator on the prey in a predator-prey relationship affects their co-evolution behaviour, using a simplified model implemented in an OpenAI Gym environment and utilizing NEAT algorithm for evolution. Results suggest that this is indeed the case, with lower selection pressure resulting in more stable co-evolution as well as higher fitness for the best individuals while higher selection pressure resulting in increased diversity of controllers. These findings may be significant, especially in the field of robotics where the diversity of a controller may be just as important as its efficacy. In such a setting, tuning the strength of selection pressure and training using predator-prey model may result in an agent with the right balance for the given task.

References

[1] L. Van Valen, "The red queen," *The American Naturalist*, vol. 111, no. 980, pp. 809–810, 1977.

[2] C. Lambert, M. C. Smith, and R. E. Sockett, "A novel assay to monitor predator–prey interactions for bdellovibrio bacteriovorus 109 j reveals a role for methyl-accepting chemotaxis proteins in predation," *Environmental Microbiology*, vol. 5, no. 2, pp. 127–132, 2003.

[3] P. Domenici, "The scaling of locomotor performance in predator–prey encounters: from fish to killer whales," *Comparative Biochemistry and Physiology Part A: Molecular & Integrative Physiology*, vol. 131, no. 1, pp. 169–182, 2001.

[4] W. C. Kerfoot and L. J. Weider, "Experimental paleoecology (resurrection ecology): chasing van valen's red queen hypothesis," *Limnology and Oceanography*, vol. 49, no. 4part2, pp. 1300–1316, 2004.

[5] J. Haafke, M. Abou Chakra, and L. Becks, "Eco-evolutionary feedback promotes red queen dynamics and selects for sex in predator populations," *Evolution*, vol. 70, no. 3, pp. 641–652, 2016.

[6] W. M. Schaffer and M. L. Rosenzweig, "Homage to the red queen. i. coevolution of predators and their victims," *Theoretical Population Biology*, vol. 14, no. 1, pp. 135–157, 1978.

[7] S. DOWNES and R. SHINE, "Sedentary snakes and gullible geckos: predator–prey coevolution in nocturnal rock-dwelling reptiles," *Animal Behaviour*, vol. 55, no. 5, pp. 1373–1385, 1998.

[8] C. Adami and C. Adami, *Introduction to artificial life*. Springer Science & Business Media, 1998.

[9] T. Ray, "An approach to the synthesis of life", artificial life ii, christopher langton, charles taylor, doyne farmer, steen rasmussen, eds," 1992.

[10] S. Nolfi, "Co-evolving predator and prey robots," *Adaptive Behavior*, vol. 20, no. 1, pp. 10–15, 2012.

[11] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

[12] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Neural Information Processing Systems (NIPS)*, 2017.

[13] A. Peters and C. Lively, "The red queen and fluctuating epistasis: a population genetic analysis of antagonistic coevolution," *The American Naturalist*, vol. 154, no. 4, pp. 393–405, 1999.

[14] J. D. Van Der Laan and P. Hogeweg, "Predator—prey coevolution: interactions across different timescales," *Proceedings of the Royal Society of London. Series B: Biological Sciences*, vol. 259, no. 1354, pp. 35–42, 1995.

[15] M. H. Cortez and J. S. Weitz, "Coevolution can reverse predator–prey cycles," *Proceedings of the National Academy of Sciences*, vol. 111, no. 20, pp. 7486–7491, 2014.

[16] P. A. Abrams and H. Matsuda, "Fitness minimization and dynamic instability as a consequence of predator–prey coevolution," *Evolutionary Ecology*, vol. 11, no. 1, pp. 1–20, 1997.

[17] D. Cliff and G. F. Miller, "Tracking the red queen: Measurements of adaptive progress in co-evolutionary simulations," in *European Conference on Artificial Life*, pp. 200–218, Springer, 1995.

[18] D. Cliff and G. Miller, "Coevolution of neural networks for control of pursuit and evasion," *University of Sussex, UK [Online]. Available: http://www. cogs. susx. ac. uk/users/davec/pe. html*, 1996.

[19] J. R. Koza, "Genetic evolution and co-evolution of computer programs," *Artificial life II*, vol. 10, pp. 603–629, 1991.

[20] J. R. Koza and J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*, vol. 1. MIT press, 1992.

[21] G. Buason, N. Bergfeldt, and T. Ziemke, "Brains, bodies, and beyond: Competitive co-evolution of robot controllers, morphologies and environments," *Genetic Programming and Evolvable Machines*, vol. 6, no. 1, pp. 25–51, 2005.

[22] G. Búason and T. Ziemke, "Competitive co-evolution of predator and prey sensory-motor systems," in *Workshops on Applications of Evolutionary Computation*, pp. 605–615, Springer, 2003.

[23] A. L. Nelson, E. Grant, and T. C. Henderson, "Evolution of neural controllers for competitive game playing with teams of mobile robots," *Robotics and Autonomous Systems*, vol. 46, no. 3, pp. 135–150, 2004.

[24] D. Floreano and S. Nolfi, "God save the red queen! competition in co-evolutionary robotics," in *Proc. of The Second Conference on Genetic Programming*, 1997.

[25] D. Floreano, S. Nolfi, and F. Mondada, "Competitive co-evolutionary robotics: From theory to practice," in *Proc. of The Fifth International Conference on Simulation of Adaptive Behavior (SAB), From Animals to Animats*, ETH Zürich, 1998.

[26] S. Nolfi and D. Floreano, "Coevolving predator and prey robots: Do "arms races" arise in artificial evolution?," *Artificial life*, vol. 4, no. 4, pp. 311–335, 1998.

[27] L. M. Antonio and C. A. C. Coello, "Coevolutionary multiobjective evolutionary algorithms: Survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 6, pp. 851–865, 2017.

[28] J. E. Auerbach and J. C. Bongard, "Evolving complete robots with cppn-neat: the utility of recurrent connections," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pp. 1475–1482, 2011.

[29] Z. Buk, J. Koutník, and M. Šnorek, "Neat in hyperneat substituted with genetic programming," in *International conference on adaptive and natural computing algorithms*, pp. 243–252, Springer, 2009.

[30] P. Caamaño, R. Salgado, F. Bellas, and R. J. Duro, "Introducing synaptic delays in the neat algorithm to improve modelling in cognitive robotics," *Neural Processing Letters*, vol. 43, no. 2, pp. 479–504, 2016.

[31] L. Trujillo, L. Muñoz, E. Galván-López, and S. Silva, "neat genetic programming: Controlling bloat naturally," *Information Sciences*, vol. 333, pp. 21–43, 2016.

[32] M. Baer, "Coevolution of multiagent systems using neat,"

[33] J. Chen, "The predator-prey evolutionary robots system: from simulation to real world," Master's thesis, University of Amsterdam, Department of Computer Science, Faculty of Informatics, Vrije Universiteit Amsterdam, 7 2019.

[34] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[35] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[36] A. Das, S. Kottur, J. M. Moura, S. Lee, and D. Batra, "Learning cooperative visual dialog agents with deep reinforcement learning," in *Proceedings of the IEEE international conference on computer vision*, pp. 2951–2960, 2017.

[37] Z. Buk, J. Koutník, and M. Šnorek, "Neat in hyperneat substituted with genetic programming," in *International conference on adaptive and natural computing algorithms*, pp. 243–252, Springer, 2009.

[38] A. McIntyre, M. Kallada, C. G. Miguel, and C. F. da Silva, "neat-python." https://github.com/CodeReclaimers/neat-python.