Computer Science and Information Systems 18(4):1175–1196 https://doi.org/10.2298/CSIS201122022V

Cooperation and Sharing of Caught Prey in Competitive Continuous Coevolution Using the Predator-Prey Domain

Krisztián Varga[1] and Attila Kiss[2]

[1] Eötvös Loránd University
Budapest, Hungary
scout@inf.elte.hu
[2] Eötvös Loránd University department of Information Systems
Budapest, Hungary
kiss@inf.elte.hu

Abstract. Competition is one of the main driving factors of evolution and can be observed in nature as well as in simulations. Competition can occur between predators and preys causing an arms race, but it can also happen between individuals of the same species. Our simulation uses the predator-prey domain (with carnivores, herbivores and plants,) and continuous (not generation based) neuro-evolution to create a complex environment where both forms of competition arise. The characteristics of the simulation make it hard for a predator to catch prey alone, this creates a dependence on cooperation. The predators can share the caught prey with nearby members, in order to help them work together. We explore how sharing affects the cooperation of the hunters, and compare the effectiveness of one and two predator populations.

Keywords: artificial life, neural networks, evolution, coevolution, cooperation, arms race, competition, predator, prey.

1. Introduction

Evolution is a critical part of nature that makes complex behaviours and higher level lifeforms possible. A lot of reasons can lead to evolution including environmental changes [11], competing species [8], and competition between individuals [3]. A relevant example today for the first one is the ongoing climate change, which has already shown its effects in nature: a 2 degrees Celsius increase in spring temperature over 10 years caused a red squirrel population in the southwest Yukon, Canada, to have an 18 days advancement in the timing of breeding [22]. It has been shown that competitive arms races arise in nature (bats and moths [12], whelk and Mercenaria [4]) as well as in simulated environments [20] [23]. It is important to study these processes to better understand one of the main driving factors of evolution.

We were interested in the performance and behaviour of simple agents that have limited knowledge about their local surroundings in a competitive and changing environment, where cooperation is key for survival. Studies confirm [5] [14] that even microbiological lifeforms usually cooperate by releasing enzymes into the common space, which helps everyone to get more nutritions. But can such low level lifeforms really cooperate? Marshall [17] who studied the "wolf-pack" hunting of myxobacteria disagrees. He argues

that they do not necessary cooperate with each other, because more bacteria indeed helps killing prey more effectively, but the released nutritions must be shared. Since such low lifeforms have no concept of teamwork and can only sense other individuals near them, the "cooperating" behaviour might only come from the loose definition of cooperation. The experiment proposed here is similar to the strategy of these microscopic predators, but instead of chemical warfare, they have to catch prey by surrounding it.

There are a number of hypothesises we want to examine. The first one is that arms race can be sustained in our continuous simulation without distinct generations. The second is that two smaller hunter populations will be more effective at catching prey, because they can specialize two different roles to capture prey and there is a possibility of arms race between the two populations. The last hypothesis is that a higher sharing percentage will increase cooperation between hunters.

2. Related Work

2.1. Evolutionary Computation

In computer science evolutionary computation refers to a family of algorithms that are used for solving global optimisation problems. They start with a population of solutions, which are usually randomly generated, and evolve them based on natural selection, crossovers and mutations. Natural selection means survival of the fittest, therefore there is often a fitness function that evaluates every solution and then the best performing ones will be selected for crossing or mutation. Crossing two or more population members means that the result will contain parts of all of the parent members, this helps to faster converge to the optimal solution. Mutation is also an important part of the process, by having a low chance for altering small parts of members we can ensure that we do not get stuck in a local optima. Evolutionary computation have led to interesting discoveries and showed solutions that exceeded the researchers expectations [15].

2.2. Neuro-evolution

Neuro-evolution is a subset of the evolutionary algorithms. It is a form of artificial intelligence, which focuses on training artificial neural networks. The training can involve the weights of the network, the structure of the network or both, in this paper we are using the first one. Artificial neural networks are often trained by a form of stochastic gradient descent algorithm, but one of the greatest advantages of neuro-evolution over them is that it does not need training sets of correct input-output pairs. It is a powerful tool for solving problems, where the system or environment is highly complex, for example controlling autonomous agents [2], rockets [6] or freeway traffic [27], which is exactly what we need.

2.3. Coevolution

Coevolution in evolutionary computation means that multiple agents belonging to two or more populations are evolving while interacting with each other and this is represented in their fitness evaluations. Coevolution can be cooperative, competitive or both at the same time. In cooperative coevolution [16] agents work together to achive a common goal and

they share a common fitness evaluation. In competitive coevolution [13] [24] agents are working against each other and one's gain in fitness means loss to it opponent(s). A good example of competitive and cooperative coevolution is Balch's work [1] where he trained robot soccer teams with reinforcement learning. He showed that global reinforcement for the teams results in better performance and more heterogeneous behaviour, because an agent is not punished even if he did not contribute as much to the overall fitness. Teams evolved with local reinforcement show homogeneous behaviour, because they are competitive against their teammates. Another study [21], where behaviour of spotted hyenas was modeled using coevolution, showed that communication and reward sharing increased cooperation in a simple simulated environment. In this paper we wanted to achieve a mix of global and local reinforcement for hunters by introducing sharing of caught prey. This way each agent will have their own fitness value, but they can benefit by working together. Another important aspect of coevolution is that it often leads to arms races between the populations and evolutionary trade-offs can also be discovered [8]. Evolutionary trade-offs are adaptations, where one feature or behaviour of an agent becomes better while other areas experience decreased performance.

2.4. Predator-Prey Domain

Predator and prey systems have been extensively studied by many scientific fields, including biology [25], mathematics [30] and computer science [7] [13]. In predator prey systems there are usually two populations where the predators' goal is to eat preys. Different aspects of this system can be analysed depending on the simulation or the real life ecosystem. One of the most studied concept is coevolution between predators and preys [13], because the model fits perfectly with its two distinct subpopulations, where one's gain is the other's loss.

Our study was inspired by a simulation [23] that was used to create competitive and cooperative behaviours using coevolution. In this study they used 3 predators and 2 preys and they managed to keep up an arms race between cornering and fleeing strategies. For the neuro-evolution a multi agent ESP [29] architecture was used, where the hidden neurons inside a network come from different sub-populations and each network inside one agent contains its own neuron sub-populations. They used multiple networks to keep track of all of the enemies' coordinates and combined them by summing their output neuron values. With this structure they created multilevel cooperation and competition. Cooperation between the hidden neurons in a network, cooperation between the neural networks inside agents, cooperation between agents of the same kind, and competition between hunters and preys.

3. Rules and Mechanisms of the Simulation

We wanted to scale up the number of agents in the simulation and to run it continuously, without distinct generations, but still benefit from evolution. The original architecture was not suited for our needs, because every agent had global knowledge of all of their enemies' positions and by scaling up to hundreds of agents one individual's movement would require hundreds of neural networks to cooperate. The spatial environment was a good starting point, because spatial coevolution proved to be effective [18], [19], [26] in

overcoming the main arising problems in coevolution: loss of gradients, (which means that one population is either too weak or too strong for the other for meaningful change to occur), over-specialization, (which means that the evolutionary process gets stuck in a local optimum), and red queen dynamics, (which means the populations continue to change, but these changes do not force more general solutions). To avoid extinction we used populations with fixed sizes. This means that whenever an agent dies, it is immediately replaced by a new one and the same happens to plants. This does not mean that there are infinite resources, because in practice plants have to be found first, which takes time. The new agent will be constructed from multiple currently alive agents. These agents are chosen from the top performing individuals in the population, which means that they are in the top 10% based on their fitness values. (The selection process is discussed in detail in section 3.5.) The 10% threshold was chosen, because after experimenting with different values, we found this to be high enough that not only a few lucky agents will be considered and low enough to ensure that only competitive genes will be passed on. This parameter could have been set to any value, but this is not in the scope of our study. This system combined with local interactions will keep the simulation running forever and ensure that rapid as well as slow evolution will take place. Since we are using agents with no memory and there is no "Hall of Fame" to preserve the best networks of all time, the evolution only reacts to the current environment. The complexity of the environment makes it highly unlikely to reach a state that was visited before, which prevents the simulation to be stuck in a periodic repetition of states and strategies.

To experiment and to draw conclusions a suitable environment is needed. We built the simulation around the predator-prey domain, which has been studied by the scientific community for over decades. Multiple studies confirm, that arms race between predators and preys can be observed in nature and that this can be reproduced in an artificial environment. Each individual in the simulation is controlled by 3 different artificial neural networks. Fitness based selection and random mutations ensure the continuous evolution.

3.1. Playground and Walls

The artificial world consists of a 2 dimensional rectangle, where the size of each dimension can be configured. There are maps that can be loaded into the simulation. A map is a collection of walls, the simplest map consists of 4 walls around the rectangle. A wall has two points, these points' coordinates are defined on the unit square. When a map is loaded all the walls' coordinates will be multiplied by the worlds dimensions, this way maps are not tied to sizes. Walls do not move and do not have thickness. If a creature comes into contact with a wall, then it dies. Walls represent static danger to all agents, it does not change throughout the simulation contrary to dynamic danger for the preys, the moving and evolving carnivores. It adds complexity at the neural networks' level, where walls have to be considered when fleeing as well as catching prey. They can be interpreted as uncrossable environmental features, like deep canyons, fast flowing rivers or in the microscopical sense a patch of toxins. Walls create a structure to the map and separate small ecosystems. They can provide shelter against predators too, if prey can stay in close proximity most hunters will not risk bumping into a wall or scaring the prey into killing itself. This alone can create an arms race, where preys and predators try to get more and more closer to the walls.

3.2. Plants

Plants are represented as disks on the map. If a prey comes into contact with a plant, then the plant will be eaten. There is always a constant amount of plants, because when one is consumed it reappears on a random location. Predators have no effect on plants and their vision is not blocked by them in order to help them find prey more easily.

3.3. Predators and Preys

Predators and preys are very similar. Both of them have a square as shape, and both of them have 5 sensors they can use to navigate. The sensors are rays starting from the creature and going to 5 different directions. Every creature can move in 8 different directions on the plane: N, E, W, S, NE, NW, SE, SW, (N = North, E = East, W = West, S = South). They have an orientation in the direction they are moving and can only turn left, right or keep going straight. The 5 sensor rays follow these directions too, one goes where the creature is facing and 2-2 go left and right relative to that. A sensor tells the creature that in the relevant direction what kind of object can be found and how far it is. For preys the objects can be walls, plants, predators or nothing if there is nothing inside the view range. For predators the objects can be walls, preys, other predators or nothing. There is always a constant amount of both types of agents, because when a creature dies it is replaced with a new one immediately.

3.4. The Brain

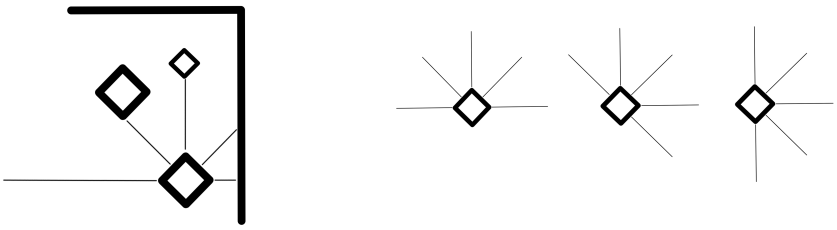
All predators and preys have a brain, which they use to decide which way they should move based on the sensor inputs. The brain is responsible for finding food and escaping static or dynamic obstacles, to accomplish this it is divided into 3 different artificial neural networks. The "food-network" is responsible for locating food, the "wall-network" avoids walls and the "predator-network" knows where predators are. For predators the food means prey, but for the preys the food means plants and another important distinction is that for the prey, predators mean danger, but for predators they are necessary companions to catch prey. The networks have the same architecture, because all of them are used the same way. The input neurons get the data from the sensor rays and the 3 outputs are the directional changes (turn left/right, keep going straight). The architecture consist of 5 input neurons, 7 hidden neurons and 3 output neurons with sigmoid activation function. Only one hidden layer was chosen, because the task is simple and more layers would have added unnecessary complexity. Since the environment needs to be optimized first (choosing default values) to be able to properly evaluate our architecture and there is no training data, we couldn't use pruning and constructing algorithms [28] [9]. There is no danger of overfitting our network, because of the lack of a training dataset and the environment will change as evolution progresses, but too few neurons could restrict our agents abilities. There is no standard way of determining the network architecture in this case, but there are some rule of thumbs that we can use [10]:

– "The number of hidden neurons should be between the size of the input layer and the size of the output layer."

– "The number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer."
– "The number of hidden neurons should be less than twice the size of the input layer."

These rules work to some extent, but every task is different and the chosen architectures have to be tested. We choose the highest number of neurons where while using 100 plants and 300 agents the simulation still runs smooth on the test machine. Although our decision makes output calculations more resource intensive, this way our agents will not be hindered by too few neurons. After the environment was balanced using this architecture, the configuration performed well during testing and allowed complex behaviours to emerge. Finding the most optimal number of hidden neurons will be a future research.

The different networks inside a single agent only get the relevant inputs for them (for example the wall-network is only concerned about detected walls). The value for a given input neuron is 0 if in the given direction there is no relevant object or $v - d$ where v represents the maximum view range and d represents the distance of the detected object. The 3 outputs represent the 3 decisions the creature can make: turn left, right, or keep going straight. The outputs of the 3 different networks is summed together and then the decision with the maximum summed weight will be selected. This creates an internal cooperation between the networks, because they are all necessary to survive and they share the host's fitness. (The fitness function is discussed in the next section.) The outputs could have been combined other ways, for example with another neural network, but this would add additional complexity to our simple agents and this method already proved to be sufficient [23].



(a) Detection of objects with rays

(b) Behaviour of rays when turning right two times

Fig. 1. Ray mechanics

3.5. Selection and Mutations

Selection occurs within each population. Since the number of agents is constant in the populations, when a creature dies another takes its place. This new agent will have 3 different neural networks and a starting position, all of these features come from other agents. The donor agents come from the top 10 percent of the population based on their fitness value at the time the replacement occurs. The randomly selected individuals from the elite fitness group are not necessary different, for example 2 or 3 networks can be inherited from the same agent. The new position will be the average of the position of an

elite agent and a random coordinate, the average will be weighted 2 to 1, this way the new agent will spawn in the near of the selected high performing teammate. This will ensure that reproduction will also happen locally, but with enough variety to not get stuck in one place.

The fitness value is calculated by how long an agent has been alive combined with how many plants or preys they have consumed. An agent starts out with a predefined amount of health. Every timestep, (the smallest time interval in the simulation or in other words one update of the artificial world,) the health declines, and if it reaches zero then the agent dies. For every timestep it survives, it gets 1 fitness point. Hunters and preys can extend their lifespan by consuming food, for preys every plant eaten will give them a predefined amount of fitness and health points, for predators every caught prey gives them a fitness score boost based on the fitness of the prey. If they catch one with a high fitness score, then they will get more fitness and health themselves for defeating a successful opponent. This is called "competitive fitness sharing" [24] and it helps avoiding stagnation.

3.6. Parameters

The simulation can be configured using a configuration file with many different parameters:

- screen_size_x: size of the x axis in pixels
- screen_size_y: size of the y axis in pixels
- updates_per_second: Simulation updates per second
- food_amount: Amount of plants on the field
- herbivore_amount: Amount of preys on the field
- carnivore_amount_1: Amount of hunters in the first group on the field
- carnivore_amount_2: Amount of hunters in the second group on the field
- map: Map file to load
- seed: Random seed, so the simulations can be repeated
- herbivore_speed: Speed of the preys
- carnivore_speed: Speed of the hunters
- initial_herbivore_health: Starting health when spawning new prey
- initial_carnivore_health: Starting health when spawning new hunter
- food_nutrition: Nutrition (health and fitness) the prey gets when it consumes a plant
- herbivore_nutrition: Base nutrition value (health and fitness) the hunter gets when it consumes a prey
- threshold_herbivore_score: The threshold fitness score for caught prey, in case of a successful hunt the nutrition the hunter gets depends on the preys fitness (new fitness = old fitness + prey nutrition * prey fitness / threshold fitness)
- sharing_percentage_1: Amount of food a hunter in the first group is willing to share if other hunters are nearby (in percentage)
- sharing_percentage_2: Amount of food a hunter in the second group is willing to share if other hunters are nearby (in percentage)
- share_range: The sharing range in pixels from a hunter that caught prey, other hunters in this range will get an equal share from the prey (the part that the hunter was willing to share will be divided equally between the others)

- mutation_rate: Probability of mutation
- herbivore_size: Size of the preys in pixels
- carnivore_size: Size of the hunters
- thinking_time: Number updates after the agents make their next decision
- view_range: Length of the sensor rays in pixels
- start_recording: Elapsed time from the start of the simulation to start recording (in minutes)
- recording_duration: Elapsed time from the start of the recording to end recording (in minutes)
- record_all_details: Recording all details of the simulation or just the events and averages (true or false)

The planning for the default simulation values started with the map. We decided that the size should be 800 by 800 pixels, because this fits on most modern monitors regardless of orientation. The map can be seen in *Figure* 2. This configuration was chosen, because there are vast spaces as well as corners where prey can hide. The prey's diameter is half the size of the hunter's, this helps prey to escape through small paths between hunters and makes hunters easier to detect. The hunters benefit from their increased size, because they can cover more area of the map. The diameters were set to be 40 and 80 pixels and the initial plan was to have 100 plants, prey and hunters in the simulation, because a population of 100 individuals is large enough for the algorithm to not get stuck in a local optima. The population of hunters had to be reduced to 90, because they took up too much area of the map and blocked advanced strategies from emerging. The prey are slightly faster than the hunters (0.8 vs 1.2 pixels/update), in order to make them hard to catch for one hunter alone and still make it possible using teamwork. At first giving the agents a lot of initial health seemed to be a good idea to make them last longer and to have enough time to find food. This worked well when there were only prey and plants on the map, but when hunters were introduced their behaviour changed. The prey were hiding from the predators and not catching any plants, making it harder for both populations to evolve. By reducing their health to 1000 and setting the reward for one plant to 200 they tended to seek food even when danger was introduced. On the other hand hunters needed as much initial help as they could get, therefore their health was set to 5000. More health equals more time to find teammates and hunt together, but if we would set it any higher they wouldn't explore the map, because it is more beneficial to them to just wait out their lifetime than to accidentally die early by colliding with a wall. The base reward for catching prey is set to 200, but the final reward depends on the fitness value of the caught agent. An agent is considered average, if it reaches 750 fitness, this means that it survived three quarters of its initial lifespan or that it managed to eat some plants. The exact reward value comes from the formula: agent's_fitness_value / 750 * 200. This ensures that killing a skillful individual is rewarded and that agents who are results of bad mutations or crossovers do not feed the hunters too much. There are two more important parameters left, the view range and the share range. The view range was not limited at first, but after a short time of running the simulation this resulted in reduced movement, by decreasing its value the agent were more keen to explore the map. With 190 agents on a map with 640000 pixels, there is approximately a 60 by 60 pixels square for each one, therefore a view range of 50 pixels is sufficient for them to be aware of their surroundings. The sharing range comes from the view range, an agent should only get a share of the price

if it contributed to the hunting process, which means that it was close when the prey died, therefore the sharing range should be less than then view range. When the sharing range was too low (10-30 pixels) the hunters did not have enough room to corner prey and still be close enough to get a share, in the end 40 pixels showed the best results.

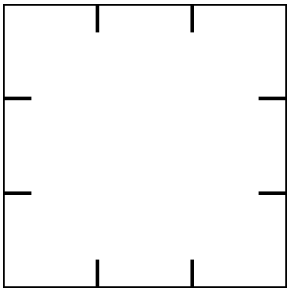


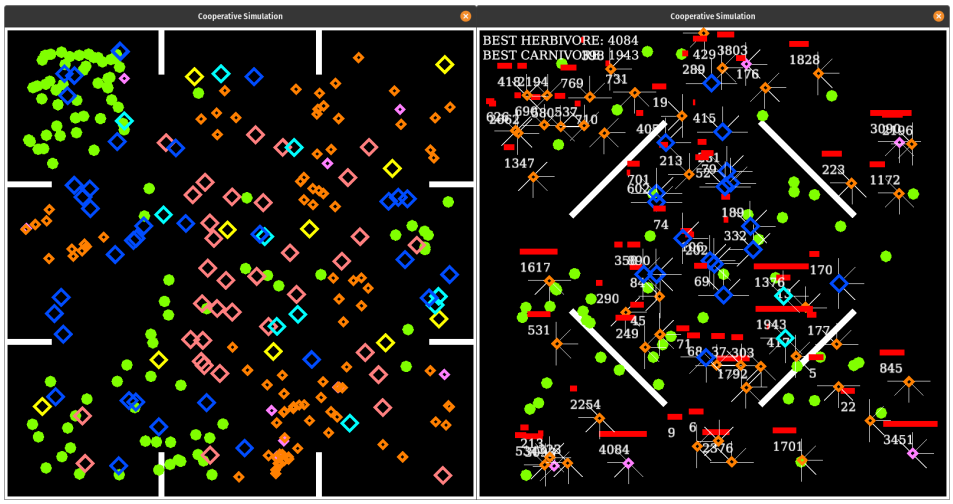
Fig. 2. The map used for the simulations

4. Experiments

The simulation was implemented in Rust, a fast and high level language that has many safety guards for memory management and concurrent programming. The code is open source and it is published on github[3]. The program does not have huge system requirements, but the performance will depend on the number of agents used. The experiments were carried out on a 6 core 3.6GHz processor. If simulation data is being recorded, we advise using an SSD.

There are a lot of parameters in this simulation that can be adjusted, therefore we had to come up with an experimenting plan. After the default values have been chosen for the parameters that we will use for all simulation runs, (more details in the *Parameters* section,) we focused on two main factors: the number of predator populations (1 or 2) and sharing percentages (0, 25, 50, 75 or 100). When there are two predator populations we can also test different sharing percentages for them (0-100, 20-80, 40-60). This gives us $5 * 2 + 3 = 13$ kinds of simulation runs. All simulations are run for 8 hours (or approximately 700000 simulation updates). The results shown here are individual runs, not averaged over multiple simulations from the same kind, because each run is different with huge spikes in the graphs at different places and we do not have the necessary time and resources to run enough simulations to smooth the graphs out.

[3] https://github.com/kvarga-research/Competitive-and-Cooperative-Evolution



(a) Agents inner state is hidden **(b)** Agents inner state is visible

Fig. 3. Screenshots of the simulation. The visibility of the agents' inner state and detection rays can be toggled. The smaller squares are preys, the bigger ones are predators. The first hunter group is red and their elite agents are yellow. The second hunter group is blue and their elite agents are cyan. The elite preys are purple.

4.1. Results

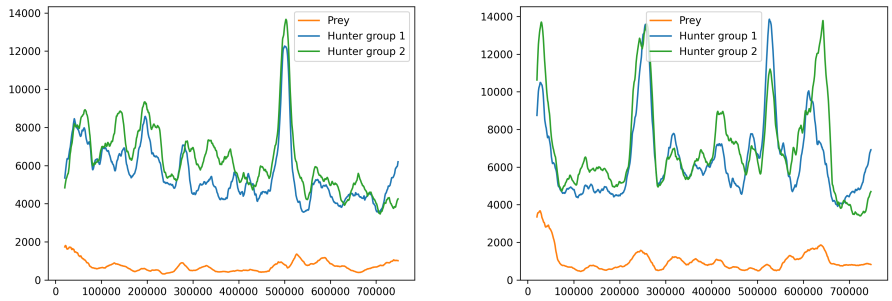
The 13 runs with different configurations will be discussed in this section. To avoid repetition of words, we introduce a naming system for the different configurations. 'PNSM' means that N population(s) was/were used with M% sharing. For example P2S50 means two populations with 50% sharing. If there are different sharing percentages, then the additional percentage will be added to the end: P2S20S80. All graphs were smoothed out with rolling average over 20000 timesteps.

The results for P2S25, P2S50 and P2S75 can be seen in *Figure* 5. The graphs on the right side show the cooperation between hunters and the first thing we can notice is that hunting involving only one individual is always the smallest and mostly 2 to 3 sized hunting groups dominate. Its interesting to see that the performance of the two hunter groups are very close throughout the entire run in P2S25 and P2S50, the green and blue lines follow each other closely, this means that they have been working and moving together, reacting the same way to the environmental and behavioural changes. If we take a look at *Figure* 4, we can see that there is an arms race between the two hunter populations. because their average fitness scores are taking turns at having the most prey captured. Looking at the P2S75 run we can see that green group is clearly more successful than the blue one, but the spikes in the graph are at the same places, this most likely means that the two teams are cooperating in a way: they form huge packs (which is beneficial when there is 75% sharing), but the green team is the one actively catching preys meanwhile the blue team might be blocking fleeing paths, or just simply having a parasitic relationship with the greens. The cooperation graph confirms this theory, because

the “equal or greater than 5” sized groups are massively dominating. These are most likely much bigger groups than 5, because if they were near 5, then statistically the 4 sized groups’ graph should be closer to it. The plant eating is much lower than the prey catching throughout the simulation, but this does not mean that they are suppressed, this only tells us that eating plants is not worth it that much. Another interesting phenomenon that we observed in most of the simulations is that since prey are running from hunters, there are areas where hunters are dominating, therefore at such places the spawned plants will not be eaten. This leaves much fewer plants for other areas and when hunters move after prey, they free up plant-rich fields, which explains periodic bumps in plant eating and prey capture.

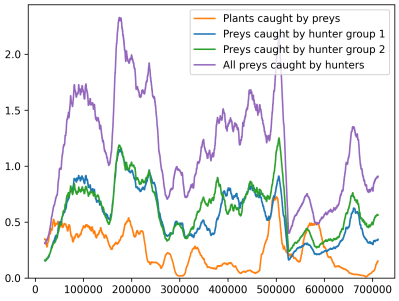
The results for P1S25, P1S50 and P1S75 is shown in *Figure* 6. The main difference compared to the “two population” version is that the overall hunter performance is worse. The performance also declines the more sharing is introduced, this is most apparent with 75% when the average preys captured are a third of the multi population counterpart. Such low number of caught prey is caused by the hunters forming big “wolf packs” with lot of agents on a very small area. These high density groups leave a lot of free space for preys to roam around and make it relatively easy to evade them. With such good conditions for preys we could expect the plant eating to skyrocket, but this is not the case, because as mentioned before, these hunter rich areas are hogging plant resources. Preys might seem poorly performing looking at the average plants eaten, but keep in mind that they have to balance eating and avoiding predators. When predators are truly dominating, the number of caught prey increases meanwhile the plant consumption drops to almost zero, because preys are eaten so fast they have no chance of finding plants. A spectacular example can be seen of this with 50% sharing between the 350000th and 400000th timesteps.

The cooperating diagrams are mostly dominated by hunter groups of size 2 or 3, but size 4 and 5 are not far behind and successful solo hunting remains almost nonexistent. These ratios are not changing much during simulations and they do not seem to be affected by different sharing levels.

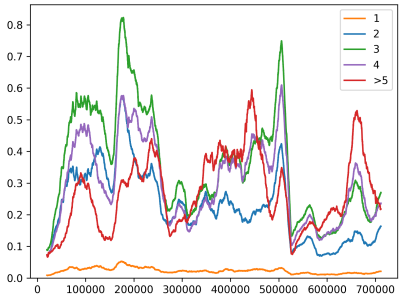


(a) 25% **(b)** 50%

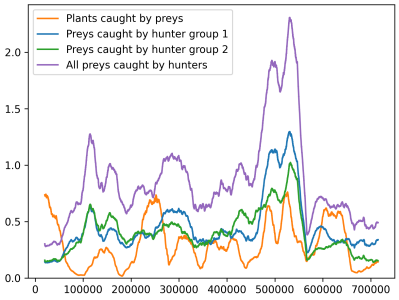
Fig. 4. Average fitness of the top 10% with 25% and 50% sharing



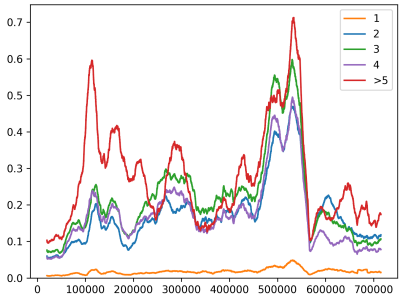
(a) 25%



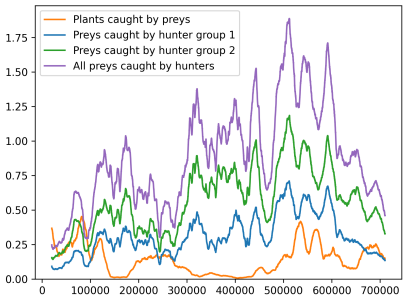
(b) 25%



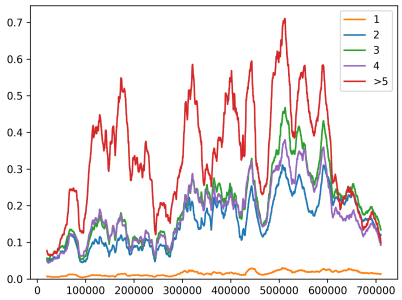
(c) 50%



(d) 50%

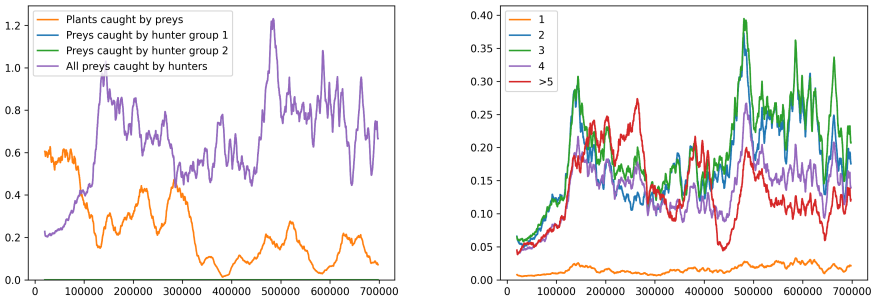


(e) 75%



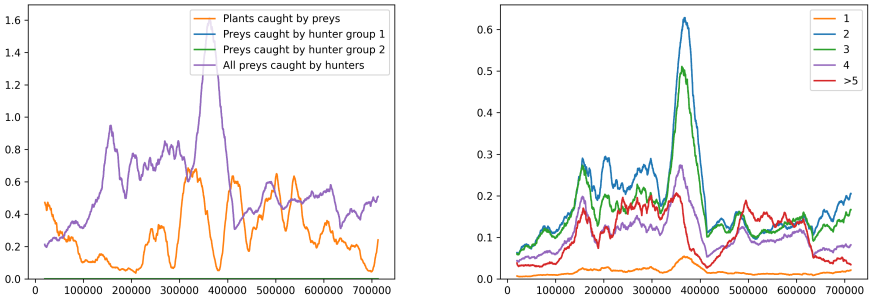
(f) 75%

Fig. 5. Successful hunting and teamwork comparison between 25, 50 and 75 percent sharing with two predator populations



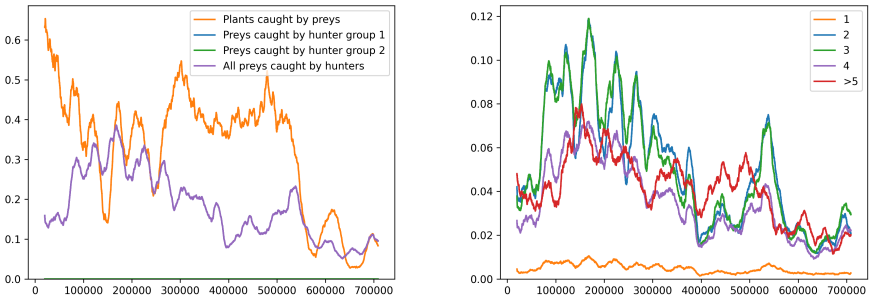
(a) 25%

(b) 25%



(c) 50%

(d) 50%



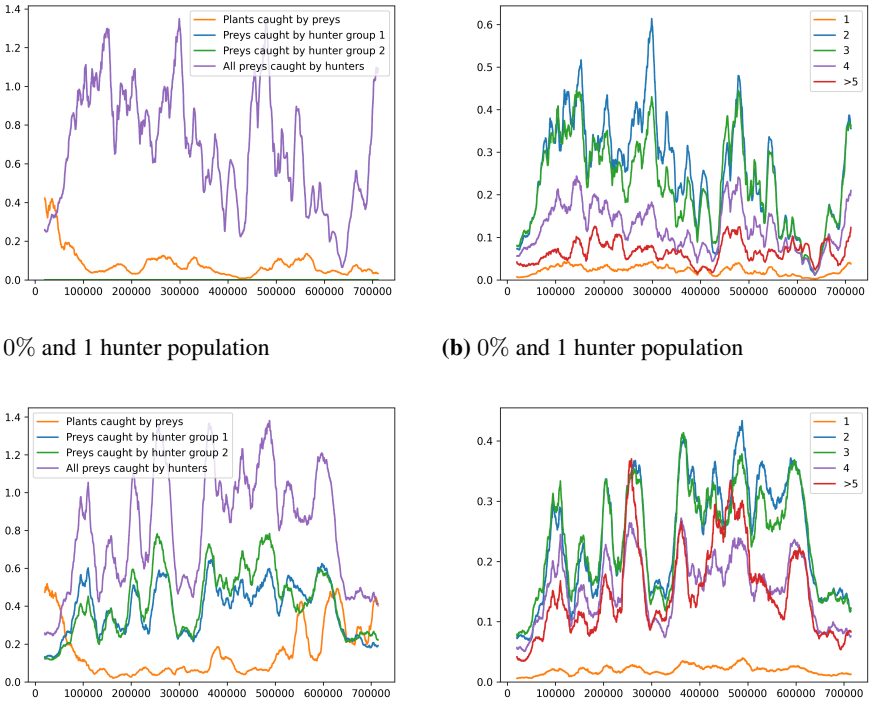
(e) 75%

(f) 75%

Fig. 6. Successful hunting and teamwork comparison between 25, 50 and 75 percent sharing with one predator population

Sharing can be also set to 0%, which means that hunters will only get rewards if they are the ones catching the prey and there is no compensation for teamwork and blocking escape paths. The corresponding simulation runs are shown in *Figure* 7. This configuration often makes hunters avoid each other as much as possible to maximize their chance

at being the one actually eating prey. This results in homogeneous hunter density over the map, which explains the consistently low amount of plant eating, because hunters are everywhere evenly spread out, making avoiding them priority over finding food. This behaviour can be observed best with one population, where all agents use very similar strategies and therefore hunters are distributed uniformly across the map. This affects co-operation by favoring smaller hunter groups, which is clearly visible in *Figure* 7/(b). The only exception is solo hunting which is still the lowest. There are two reasons for this, first of all hunting alone is a difficult task and prey caught this way probably has a low amount of fitness, which means that the hunter will not get enough nutrition out of it to survive. The second reason is the even distribution of hunters that makes it hard to be alone while hunting. With two populations the results are not as organized, but smaller groups are dominating in this case too with a few exceptions time to time. Once again with multiple populations the performance of the two hunter teams are closely following each other and their arms race is observable by examining the health and fitness of their top performing agents in *Figure* 8. The overall hunter performance is similar with both one and two populations and compared to previous configurations it is at the same level as P1S25 and P1S50.



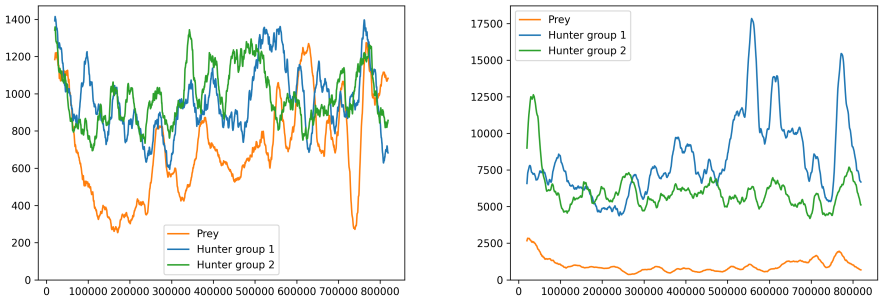
(a) 0% and 1 hunter population

(b) 0% and 1 hunter population

(c) 0% and 2 hunter populations

(d) 0% and 2 hunter populations

Fig. 7. Successful hunting and teamwork comparison with 0% sharing



(a) Health **(b)** Fitness

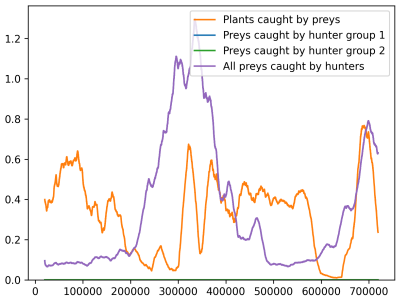
Fig. 8. Average health and fitness of the top 10% with 1 population and 0% sharing

Another special case is when there is 100% sharing. This means that whoever catches a prey will not get anything out of it, because it will be divided between other hunters nearby. If there is no one else inside the sharing radius, then the solo hunter gets all the nutritions. Even with this huge benefit for hunting alone, the results in *Figure* 9 show that this is still the least common method of successful hunting. With one population there is one huge spike in hunting from the 200000th time step to the 500000th, then at 600000 another spike starts growing. It looks like the hunters came up with a strong strategy to catch prey, but this is not the case. If we take a look at *Figure* 10 we can see that even though hunting numbers are high, a superior strategy is not reflected in the fitness of the best performing hunters. This can occur when preys are all easily caught and they do not prioritise avoiding enemies, therefore they can only provide small amounts of nutrition. The two small spikes in hunters' fitness correlates to the two spikes in plant eating at 320000 and 380000.

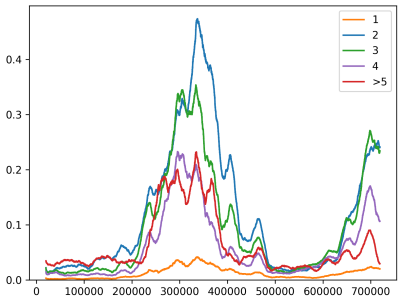
An interesting phenomenon can sometimes occur in these simulations, because of the locality of reproduction predators can force all preys into a corner and surround it, but not eat all of them at once. Given these circumstances plants quickly run out in the area and any attempt at escape is impossible. This produces high amount of capture and weak preys not providing enough energy. The situation can be broken up either by a successful escape attempt or by chance, when the new spawn point is generated behind the enemy lines.

In the case of two populations we can see a steady increase in prey capture throughout the whole simulation. In the first part huge hunter groups are dominating, but later smaller groups with 2 or 3 predators start to rise higher and higher, meanwhile the hunts including large groups are stagnating. At first the hunters were not successful, this led to preys eating a lot of plants, but as the evolution progresses this completely changes to the opposite. It looks like a gradual overtaking from the predators, but their progress was set back multiple times, the most noticable at the 380000th timestep and a smaller one at the 530000th. The recorded data of simulation did not tell us anything of these events, so we had to rely on

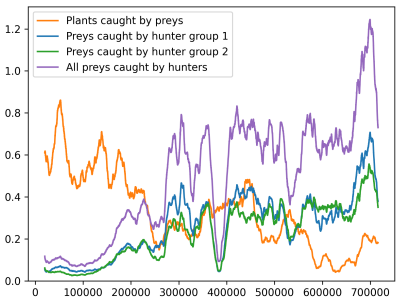
our own observations. During these periods preys became more and more cautious and avoid predators further, meanwhile predators did not spread out leaving space to do so. Such a huge drop could have caused the extinction of predators, but in our system they recovered and started catching prey again.



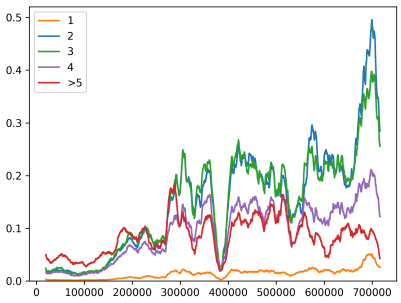
(a) 100% and 1 hunter population



(b) 100% and 1 hunter population



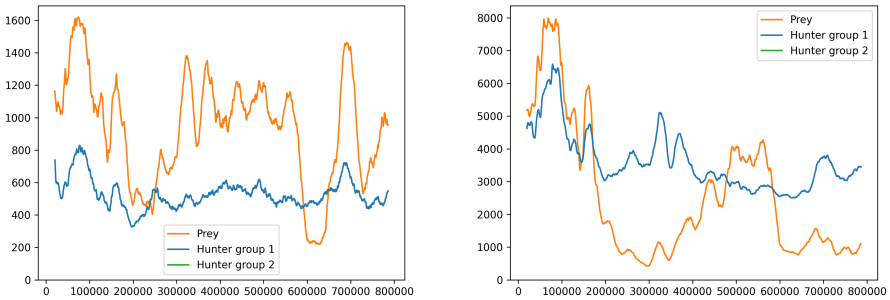
(c) 100% and 2 hunter population



(d) 100% and 2 hunter population

Fig. 9. Successful hunting and teamwork comparison with 100% sharing

When we run simulations with two predator populations their sharing percentages can be set to be different. The results for P2S100S0, P2S80S20 and P2S60S40 is shown in *Figure* 11. P2S100S0 has the most diverse population, it consist of selfish and selfless agents. Surprisingly, the selfish population worked harder and caught more prey, most likely because the reward was high and it did not needed to be shared. Members of the selfless population can only get nutritions if another selfless agent hunts prey near them, this was not enough motivation for hunting. At first bigger groups were more common, but as the simulation progressed groups of size 2 and 3 took over. This change separated behaviours in the two populations. The smaller groups performance aligns almost perfectly with the selfish team's performance, and success of groups with 4 or more hunters aligns with the selfless team.



(a) Health **(b)** Fitness

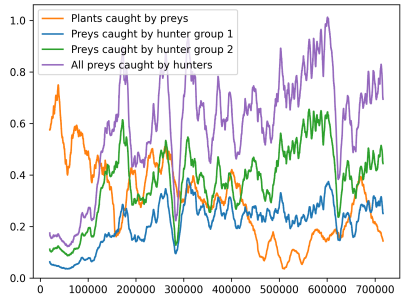
Fig. 10. Average health and fitness of the top 10% with 1 population and 0% sharing

With configuration P2S80S20 the selfless population is still less successful at first, but it closely follows the other teams performance and eventually it takes the lead. Overall the hunters have a higher efficiency than in P2S100S0, similar to P2S25, P2S50 and P2S75. Interestingly, huge groups were not as common as expected with half the population having high amount selflessness, teams containing 2 or 3 predators dominated.

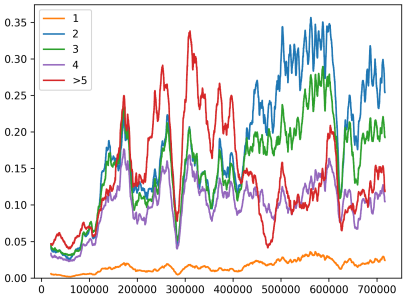
One would expect that configuration P2S60S40 is very similar to P2S50, but the results tell otherwise. The overall performance is similar to P2S100S0 and there is visible arms race between the two hunter teams. A more interesting dynamic can be spotted between the hunters and preys, from 330000th to 410000th timestep the preys valued catching plants more than running from predators. This resulted in a spike in both plant eating and prey catching and then when the priority focused again on survival, both of them dropped significantly. Unlike with P2S50, here large hunter groups were not dominant at first, but this changed when the prey behaviour changed and a new strategy was needed to catch them.

5. Summary

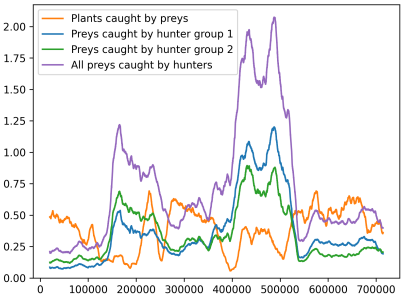
So far we have discussed the individual runs and the hunters ability to catch prey. However, the agents were not trying to maximize the amount of food caught, they were trying to survive including eating and avoiding dangers, and their fitness values represent this. The simulations are diverse regarding the agents' average fitness at any specific timestep, but we needed to come up with a way to represent them easily for easier comparison between them. To measure the performance of the predators, we calculated the overall average of the predators' fitness score averages at every timestep. This was done twice, once with every hunter included, and then with only best performers (top 10%) at every timestep. To visualize the cooperation between hunters, we calculated the average number of hunters that played a role in a successful prey capture. The results for the simulations can be seen in *Figure* 12.



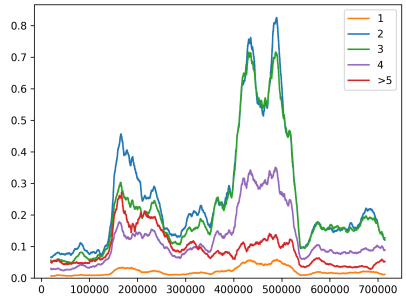
(a) 100% and 0% sharing



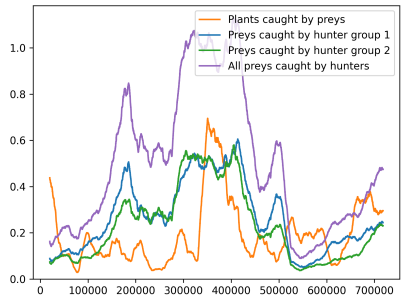
(b) 100% and 0% sharing



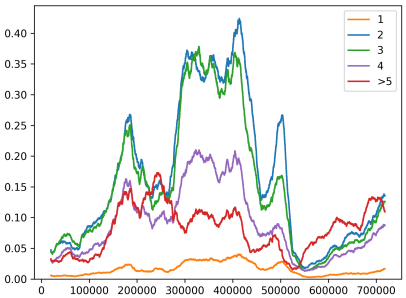
(c) 80% and 20% sharing



(d) 80% and 20% sharing

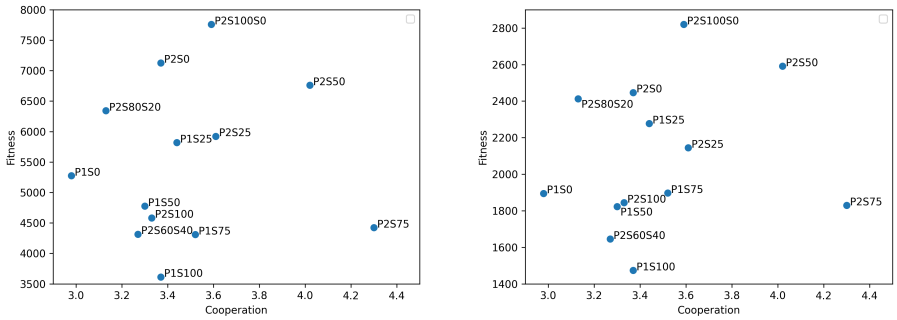


(e) 60% and 40% sharing



(f) 60% and 40% sharing

Fig. 11. Successful hunting and teamwork comparison between simulations, where populations have different sharing percentages



(a) Best performing hunters (top 10%) only **(b)** Fitness of all hunters

Fig. 12. Comparison of different configurations. The fitness is the average of the average fitness values of the hunters in every timestep. The cooperation is calculated by averaging the number of hunters in every successful hunt during the simulation. The results where the fitness was calculated only using the best hunters in every timestep can be seen in (a), the results for all hunters is shown in (b).

First take a look at the configurations with 1 population. The extreme case P1S0 is an outlier in performance as well as in cooperation. All others have a cooperative score at around 3.4 while P1S0 falls behind significantly and all the others follow the rule: more sharing equals less fitness. Following this rule P1S0 should have had the best fitness performance, but it only came in second. Comparing the 1 and 2 population results we can see that configurations with 2 populations always performed better regarding their fitness, often by a significant margin. Looking at the cooperation values it is the same situation, with only one exception (P1S100 and P2S100, which are both extreme cases). Focusing on only those with 2 populations and homogeneous sharing we can see that more sharing helps cooperation, again with one exception P2S100. Their fitness scores are not so consistent, overall with more sharing the performance decreases, but this is noticeable only with high sharing values (P2S75 and P2S100). The last three configuration with heterogeneous sharing show that the more diverse the population is the better the performance, however they are not consistent compared to their homogeneous counterparts and they have under average cooperation values. They do not fit into the picture painted by all the other configurations, the reason for this could be that more diverse populations have more potential to be volatile and our sample size was too small for them.
Overall our results suggest that a complex environment with a hybrid type of reinforcement (local for the individual agents, but with more ”global” (involving multiple agents in an area) rewards for cooperating) has a significant impact on performance and cooperation. Contrary to Balch’s [1] work, where his study resulted in better performance when global sharing was used, our findings show the opposite, but we have to keep in mind that our approach did not use global reinforcement for the whole population. Although performance decreased, cooperation was more common with reward sharing, similarly to Rajagopalan’s work [21]. We were also able to utilize Rawal’s [23] work and create our own scaled up environment with high number of agents, where we were able to keep up

an arms race in different configurations. These arms races often led to evolutionary trade offs and agents constantly tried to balance out feeding, avoiding danger and cooperation.

6. Conclusion

We successfully built a simulation framework where special aspects of artificial life and evolution can be observed, using cooperating neural networks in simple and memoryless agents. This was ensured by never letting the species go extinct, because instead of distinct generations a continuously changing "elite-fitness" population was used for selection and reproduction. We have shown that arms races arise in this complex continuous simulation, the behaviours of agents is constantly changing and they do not get stuck in a local optimum. We found that predators had better performance with two populations compared to one when the same amount of sharing percentage was used. We also showed that simulations with two populations resulted in more cooperation, but more sharing also decreased the fitness score. Overall the simulations never stagnated and were always changing thanks to the localized interaction and reproduction and fixed population sizes. There is also room for improvement, for example finding the optimal neural architecture or quantifying the diversity of the agents in different runs. We hope this framework will provide ground for more research in the future and helps understand rapid evolution and cooperation in competitive environments better.

Acknowledgments. The project has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002).

References

1. Balch, T.: Learning roles: Behavioral diversity in robot teams. Multiagent Learning: Papers from the AAAI Workshop (1997)
2. Beer, R.D., Gallagher, J.C.: Evolving dynamical neural networks for adaptive behavior. Adaptive Behavior 1(1), 91–122 (1992)
3. Dawkins, R., Krebs, J.R.: Arms races between and within species. Proceedings of the Royal Society of London. Series B, Biological Sciences 205(1161), 489–511 (1979)
4. Dietl, G.P.: Coevolution of a marine gastropod predator and its dangerous bivalve prey. Biological Journal of the Linnean Society 80(3), 409–436 (2003)
5. Folse, H., Allison, S.: Cooperation, competition, and coalitions in enzyme-producing microbes: Social evolution and nutrient depolymerization rates. Frontiers in microbiology 3, 338 (2012)
6. Gomez, F.J., Miikkulainen, R.: Active guidance for a finless rocket using neuroevolution. In: Genetic and Evolutionary Computation Conference. pp. 2084–2095. Springer (2003)
7. Gras, R., Devaurs, D., Wozniak, A., Aspinall, A.: An individual-based evolving predator-prey ecosystem simulation using a fuzzy cognitive map as the behavior model. Artificial Life 15, 423–463 (2009)
8. Hague, M.T., Toledo, G., Geffeney, S.L., Hanifin, C.T., Brodie Jr, E.D., Brodie III, E.D.: Large-effect mutations generate trade-off between predatory and locomotor ability during arms race coevolution with deadly prey. Evolution letters 2(4), 406–416 (2018)
9. Hassibi, B., Stork, D.G., Wolff, G.J.: Optimal brain surgeon and general network pruning. In: IEEE International Conference on Neural Networks. vol. 1, pp. 293–299 (1993)
10. Heaton, J.: Introduction to neural networks with Java. Heaton Research, Inc. (2008)

11. Hoffmann, A.A., Parsons, P.A. (eds.): Extreme environmental change and evolution. Extreme environmental change and evolution, Cambridge University Press (1997)
12. Hofstede, H., Ratcliffe, J.: Evolutionary escalation: The bat-moth arms race. The Journal of Experimental Biology 219, 1589–1602 (2016)
13. Ito, T., Pilat, M.L., Suzuki, R., Arita, T.: Population and evolutionary dynamics based on predator–prey relationships in a 3d physical simulation. Artificial Life 22(2), 226–240 (2016)
14. Kovács, Á.: Impact of spatial distribution on the development of mutualism in microbes. Frontiers in Microbiology 5, 649 (2014)
15. Lehman, J., Clune, J., Misevic, D., Adami, C., Altenberg, L., Beaulieu, J., Bentley, P.J., Bernard, S., Beslon, G., Bryson, D.M., et al.: The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities. Artificial life 26(2), 274–306 (2020)
16. Lesser, V.: Cooperative multiagent systems: A personal view of the state of the art. Knowledge and Data Engineering, IEEE Transactions on 11, 133 – 142 (1999)
17. Marshall, R., Whitworth, D.: Is "wolf-pack" predation by antimicrobial bacteria cooperative? cell behaviour and predatory mechanisms indicate profound selfishness, even when working alongside kin. BioEssays 41 (2019)
18. Mitchell, M.: Coevolutionary learning with spatially distributed populations. Computational intelligence: principles and practice 400 (2006)
19. Mitchell, M., Thomure, M.D., Williams, N.L.: The role of space in the success of coevolutionary learning. In: Artificial Life X: Proceedings of the Tenth International Conference on the Simulation and Synthesis of Living Systems. pp. 118–124. MIT Press (2006)
20. Nolfi, S., Floreano, D.: Coevolving predator and prey robots: Do arms races arise in artificial evolution? Artificial Life 4, 311–335 (1998)
21. Rajagopalan, P.: The evolution of coordinated cooperative behaviors. Ph.D. thesis, Department of Computer Science, University of Texas at Austin (2016)
22. Réale, D., Mcadam, A., Boutin, S., Berteaux, D.: Genetic and plastic responses of a northern mammal to climate change. Proceedings. Biological sciences / The Royal Society 270, 591–596 (2003)
23. Rawal, A., Rajagopalan, P., Miikkulainen, R.: Constructing competitive and cooperative agent behavior using coevolution. In: Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games. pp. 107–114 (2010)
24. Rosin, C.D., Belew, R.K.: New methods for competitive coevolution. Evolutionary Computation 5(1), 1–29 (1997)
25. Savino, J.F., Stein, R.A.: Predator-prey interaction between largemouth bass and bluegills as influenced by simulated, submersed vegetation. Transactions of the American Fisheries Society 111(3), 255–266 (1982)
26. Williams, N., Mitchell, M.: Investigating the success of spatial coevolution. In: Proceedings of the 7th annual conference on Genetic and evolutionary computation. pp. 523–530 (2005)
27. Wu, Y., Tan, H., Jiang, Z., Ran, B.: Es-ctc: A deep neuroevolution model for cooperative intelligent freeway traffic control. arXiv preprint arXiv:1905.04083 (2019)
28. Yann L. Chun, John S. Denker, S.A.S.: Optimal brain damage. Advances in Neural Information Processing Systems pp. 598–605 (1990)
29. Yong, C.H., Miikkulainen, R.: Coevolution of role-based cooperation in multiagent systems. IEEE Transactions on Autonomous Mental Development 1, 170–186 (2009)
30. Zhang, S., Meng, X., Feng, T., Zhang, T.: Dynamics analysis and numerical simulations of a stochastic non-autonomous predator–prey system with impulsive effects. Nonlinear Analysis: Hybrid Systems 26, 19–37 (2017)

Krisztián Varga received his BSc degree in computer science at the Faculty of Informatics, Eövös Loránd University in Hungary and currently doing his master's degree with the

specialization of information systems. During his studies he attended several projects related to machine learning. His research interest is focusing on algorithms, programming, artificial intelligence. https://www.researchgate.net/profile/Krisztian-Varga-2

Attila Kiss defended his PhD in the field of database theory in 1991. His research is focusing on information systems, data mining, artificial intelligence. He has more than 165 scientific publications. Seven of his supervised students got their PhD degree. Since 2010 he has been the head of Department of Information Systems at Eötvös Loránd University, Hungary. He is also teaching at J. Selye University, Slovakia.
https://www.researchgate.net/profile/Attila-Kiss-3

Received: November 22, 2020; Accepted: April 04, 2021.