DOI: http://dx.doi.org/10.7551/978-0-262-31050-5-ch045

Task decomposition with neuroevolution in extended predator-prey domain

Ashish Jain[1], Anand Subramoney[1] and Risto Miikulainen[1]

[1]University of Texas at Austin, Austin, TX 78701
{ajain,anands,risto}@cs.utexas.edu

Abstract

Learning complex behaviour is a difficult task for any artificial agent. Decomposing a task into multiple sub-tasks, learning the sub-tasks separately, and then learning to use them as a whole is a natural way to reduce the dimensionality and complexity of the task function. This approach is demonstrated on a predator agent in the predator-prey-hunter domain. This extended domain has a new agent, a 'hunter', that chases the predators. The evading and chasing behaviours are learnt as separate sub-tasks by separate networks using the NEAT neuro-evolution method. A separate network is then evolved to use these networks based on the situation. Task decomposition using this approach performs significantly better in the predator-prey-hunter domain compared to a monolithic network evolved directly on the whole task.

Introduction

Developing complex behavior using machine learning is still a challenging goal for artificial life. At a high level, many such problems have a natural solution – split the large complex task into smaller manageable parts. Solving the parts may be easier than solving the entire problem at once and these smaller solutions then can be combined to give a solution for the entire problem.

This paper presents a neuroevolution approach to such a task decomposition in a predator-prey domain with multiple hunters chasing a predator that is also trying to catch a prey. The predator-prey domain is a well studied problem in machine learning (Benda et al. (1986)). It has also been studied in several variations in the evolutionary context (Luke and Spector (1996), Miller and Cliff (1994), Haynes and Sen (1996), Yannakakis and Hallam (2005), Yong and Miikkulainen (2010), Rajagopalan et al. (2011)). Although it is not a complex real-world domain, it is a versatile domain that can be used to illustrate important concepts of problems and approaches.

There are multiple parameters that can be varied in the predator-prey domain including relative speed of the prey with respect to the predator, number of predators, number of prey, the type of the world (continuous, closed toroidal, plane etc.), having separate teams of predators and/or prey, whether both the predator and prey learn or one has fixed behaviour, etc. Additional goals may also be added to the problem apart from capturing prey. Each of these variations alters the problem significantly and also changes the difficulty of learning the problem significantly. For instance, having multiple predators, and defining the capture method as one or more predators occupying cells adjoining the prey in all directions makes the task cooperative. On the other hand, allowing only one of the predators to capture the prey at a time, and that predator receiving the entire reward for capture of the prey, makes the problem competitive. The prey may also be evolved along with the predator, leading to an arms race between the predators and the prey. The introduction of multiple agents and multiple sub-goals makes the domain quite difficult for a simple network to solve.

Neuro-evolution as a method of training neural networks has been successfully used to solve large complex domains Yao (1999), Stanley et al. (2005), Floreano and Urzelai (2000), Gomez and Miikkulainen (1997). Although computationally more intensive than back-propagation, it is less prone to stagnation and more efficient in searching complex landscapes. One of the more successful neuro-evolution techniques is Neuro-evolution of Augmenting Topologies (NEAT) Stanley and Miikkulainen (2002). NEAT evolves increasingly complex networks in each generation, starting from a very simple network. NEAT is chosen here because it evolves both the weights and the topology of the network and it tends to find a solution close to the minimal size.

In this paper, the predator-prey-hunter task is decomposed into predator-prey and predator-hunter tasks. Networks are trained using NEAT on each of these tasks. These sub-networks are combined using another selection network that is also trained using NEAT. This selection network chooses between the sub-networks given the positions of the hunters and prey in the overall task. Such a decomposed and hierarchical approach is shown to perform much better than training a single monolithic network for the overall task. It is also shown that with increasing complexity of the domain (with more hunters), this hierarchical approach outperforms the monolithic network by an increasing magnitude.

Related work on task decomposition is first discussed. A brief outline is then given of the predator-prey domain followed by a description of the approach to solving the extended predator-prey domain with multiple sub-goals using task decomposition. The experiments are described in detail and the paper concludes with the discussion and future work.

Related Work

Task decomposition has been studied before in several other domains using different training and combination methods.

Lee (1999), studied the task of finding a box in an enclosure and pushing it towards a light source by a robot, decomposing it into separate subtasks of finding the box, positioning the robot, and pushing the box in a straight line. Separate controller circuits were evolved in simulation for each of the sub-tasks, one at a time, using Genetic Programming (GP). Then higher level controller circuits were then evolved to select the appropriate sub-task controller based on the sensory inputs. Such a decomposition of the overall task into separate subtasks performed better than evolving a monolithic controller circuit. The current paper follows a similar approach but evolves neural networks using NEAT instead of controller circuits.

On the soccer-keepaway task, Whiteson et al. (2005) evolved an agent for playing keep-away. Keep-away is a subdomain in robo-soccer where one team of agents, the *keepers*, try to keep the ball away from the other team, the *takers*, within a given fixed region. The subtasks were: intercepting a pass, passing the ball to a team mate, evaluating if passing a ball to a particular team mate is viable, and moving to a good position for intercepting the ball. The agent was trained for each of these sub-tasks separately, and then all these networks were combined, using decision tree in one case, and a combiner network in the other case. Their performance was compared to the case where a single network was evolved for all four tasks simultaneously. The task decomposition gave significantly better results than having a single monolithic network. Apart from the overall performance, there were some interesting behaviours observed in the modular network that was not present in the monolithic network. In particular, the agent learnt to approach the ball from the direction opposite to that in which it was going to kick the ball, since "kicking" the ball in this domain was actually coming in contact with the ball at the right velocity. The agent actually learnt that it was inefficient to first approach the ball from an arbitrary direction, and then move to the right position to kick the ball. Task decomposition gave good results only when a fixed decision tree was used to combine the subnetworks. Evolving the combiner network didn't perform as well as the fixed decision tree. The goal of the current paper is to show that a proper combination of the subtasks enables the combiner network to be learned as well.

There has also been some work done on learning tasks incrementally (Gomez and Miikkulainen (1997) being one of them) – starting with a simple task, and slowly increasing the task difficulty as the network learns. This approach is different from the task decomposition addressed in this paper in that the task remains the same, and just a few parameters of the task are varied to make it more difficult. For instance, in the predator-prey domain, the speed of the prey is increased slowly. In contrast, the task decomposition approach in the current paper divides the task into specific subtasks and later combines them.

In Yong and Miikkulainen (2010), multi-agent ESP (Enforced Sup-Populations) was used to coevolve multiple networks for each set of inputs for a predator-prey task, and it was shown that this coevolved network performs better than a monolithic network when there were multiple predators and prey involved. This work was extended in Rajagopalan et al. (2011) to domains with different types of prey, and with individual and shared fitness, where cooperation between the agents was seen to evolve. Multi-agent ESP decomposes the overall network in terms of the inputs automatically, but cannot be directly applied to arbitrary task decomposition. The current paper develops a mechanism to decompose networks for arbitrary (manually specified) task decomposition. Currently the NEAT neuroevolution method is used, although in the future, multi-agent ESP could be modified to work for arbitrary task decomposition.

The Extended Predator-Prey Domain

A toroidal grid world of size 10×10 with one predator, one prey and multiple hunters is used. This is illustrated in Figure 1, which has four hunters (filled blue circles), one predator (red square) and one prey (black circle). The agent being evolved is the predator. The goal of the predator is to capture the prey in as few time steps as possible without being caught by the hunter(s) in the process.

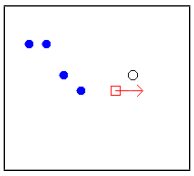


Figure 1: Illustration of the extended predator-prey domain. The four filled blue circles are the hunters chasing the predator, which is indicated by the red open square. The black open circle is the prey being chased by the predator.

"Capture" of the prey is defined as the predator occupying the same cell as the prey. Likewise, capture of the predator is defined as a hunter occupying the same cell as the predator. The behaviour of the prey and the hunters are fixed. The prey always moves away from the predators with a fixed move probability, while the hunter moves towards the preda-

tor with a fixed move probability.The prey and the hunters move slightly slower than the predator, their move probabilities being 0.8 (and hence their speed is 0.8 times the speed of the predator).

If the predator is caught by the hunters, it receives a large negative reward equal to -10 times the number of remaining steps in the episode. And hence the predator agents have to learn to stay away from the hunters, apart from chasing and capturing the prey. On capturing the prey, the predator receives a large positive reward of 10 times the number of remaining steps in the episode. If the predator neither catches the prey nor is caught by the hunter at the end of 100 steps, the episode ends and the predator receives a small positive reward equal to the difference of its distance from the hunter and its distance from the prey. This means that it receives a larger reward if it is farther away from the hunter and closer to the prey.

In this extended domain, the two tasks that the predator has to do – running away from the hunters and chasing the prey are not completely independent. If the predator were to blindly chase the prey (or blindly evade the hunter), it will not be successful. It would keep getting caught by the hunter (or not catch the prey at all), since the hunter is programmed to always chase the predator (and the prey to always run away from the predator). A successful strategy would involve doing both tasks simultaneously as much as possible, and if not, run away from the hunter, since the reward on capture by the hunter is negative. The tasks are not very tightly coupled either, in the sense that the predator does not always have to do both simultaneously to be successful. A strategy of alternation between the tasks would also work. The primary reason such a task was chosen was that (1) the behaviours for each task is easily identifiable and well defined, and (2) the tasks are neither too tightly nor too loosely coupled. It provides, in the authors' opinion, a good balance of behaviours similar to those many animals exhibit in the real world.

Method

The task of the predator agent is decomposed into two parts – capturing the prey, and avoiding the hunter. The predator agent is trained separately on each of these subtasks i.e. the agent is first trained in an environment with only one prey and no hunters where it learns to capture prey successfully. Then the agent is trained in an environment with only one hunter and no prey, where it learns to avoid the hunter successfully.

A selection network is then evolved in the presence of one prey and multiple hunters. This selection network chooses between the outputs of the one modular prey chasing network (which gets the relative position of the prey as input) and n modular hunter evading networks (each of which gets the relative position of one hunter as input). These n hunter evading networks are copies of the hunter evading network evolved in the subtask. The selection network sees the entire domain, i.e. the positions of the prey and all the hunters. The task of the selection network is to decide which agent it wishes to chase or avoid at any given time step given this information. In practice only one of two networks (the prey chasing or hunter avoiding network) has to be activated, with the relative positions of the selected agent as the input. Since the selection network selects only one task at a time, it would seem that it might have trouble doing both the evading and chasing simultaneously. But, as will be seen later, the selection network is able to switch between multiple tasks fast enough to accomplish both the tasks simultaneously, and it does it surprisingly effectively. The results of this selection network are compared with a monolithic network – a single network that is evolved to solve the entire domain (without any modularity).

Experiments

The experiments were conducted on a 10×10 toroidal grid. Since the normalized relative x and y positions of the prey and hunters are provided to the predator the effect of increasing the grid size is not significant.

A total of six experiments each were conducted for both the monolithic and the selection network. For each experiment the predator network was evolved for 200 generations. Each experiment was conducted 30 times and the results were averaged. Two hundred generations was chosen because while running initial experiments for 1000 generations it was seen that the fitness of the monolithic and selection network did not change much after 200 generations. NEAT neuroevolution was restricted to feed forward networks for simplicity, since memory was not strictly required to solve this task. The number of hunters was varied from 0 to 5 in six separate experiments. The prey chasing modular network was evolved for 1000 generations with only one prey and no hunter in the domain. Likewise, the hunter evading modular network was evolved for 1000 generations with only one hunter and no prey in the domain. At the end of each generation the champion fitness i.e. the fitness of the best performing network in that generation, was saved.

The same chasing and evading networks were used in all six experiments, i.e only the selection network was evolved in each of them.

Results

The champion fitness for experiments conducted with 1, 3 and 5 hunters are shown in figures 3, 4, and 5, respectively. The results are summarized in table 1. Figure 2 shows the mean champion fitness as the difficulty of the task i.e. the number of hunters increases.

As can be seen from figure 2, the task decomposition approach performs better than the monolithic approach in every experiment with any hunters in the world. Further, as the difficulty of the task increases (number of hunters > 3),

Champion Fitness			
Number of Hunters	Monolithic	Selection	Percentage Improvement
0	822.66	832.00	1.13
1	703.65	752.50	6.94
2	503.68	704.46	39.86
3	204.02	649.40	218.29
4	-138.69	278.80	301.01
5	-79.41	289.96	465.14

Table 1: Champion fitness of generation 200 (averaged over 30 experiments). Notice that for domains with 4 and 5 hunters the monolithic network has very low fitness and is unable to catch the prey at all on average whereas the selection network is still able to catch the prey.

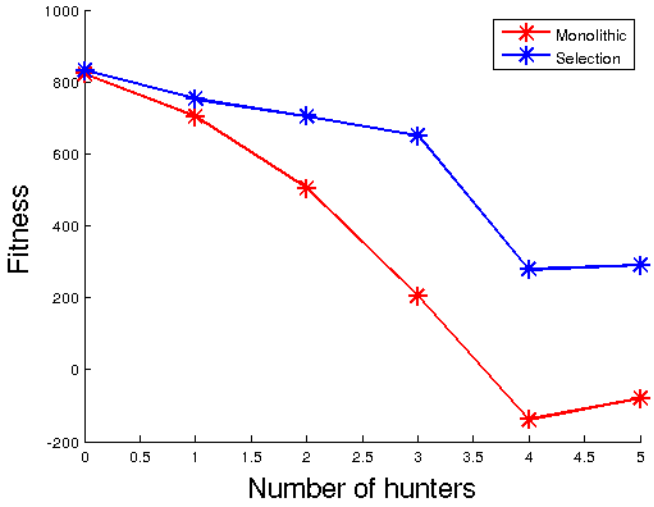


Figure 2: Average Champion fitness of Monolithic network (red) and Selection network (blue) as the number of hunters increases. The average champion fitness has been computed using champions of generation 200 averaged over 30 experiments.

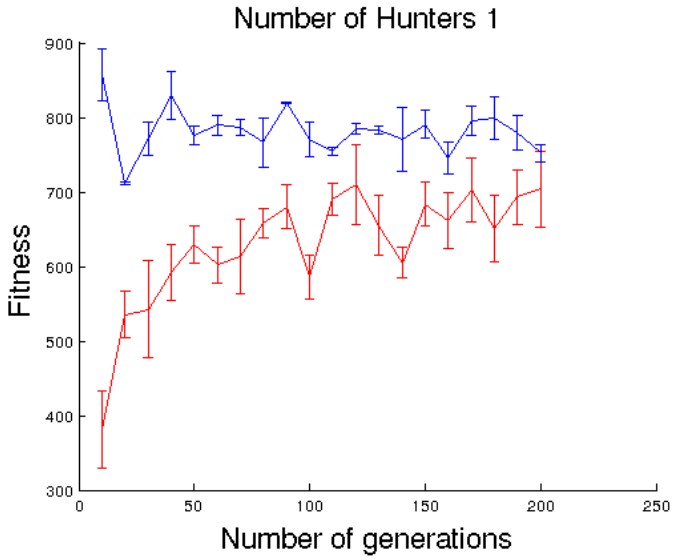


Figure 3: Champion fitness of Monolithic network (red) and Selection network (blue) with only one hunter and one prey. The results have been averaged over 30 experiments. The error bars represent the standard deviation.

the selection network does increasingly better, relative to the monolithic network. Inspection of its behavior suggests that with that many hunters, it cannot balance the two tasks, but focuses on mostly surviving or escaping from the multiple hunters. On the other hand, the selection network deals with the increase in the difficulty of the task gracefully. It is able to catch the prey even with more than three hunters although it takes more time.

Table 2 shows how the number of hidden neurons of the champion networks varies as the difficulty of the task increases. Each hidden neuron increases the number of parameters, which can be detrimental in finding the optimal solution. The selection network searches for a solution in a much smaller space compared to the monolithic network, making it easier to find good solutions, which is indeed the main benefit of task decomposition.

Behavior The monolithic network was strongly affected by hunter movements. Even though for the one hunter and two hunter case, the monolithic network was overall focused on the prey, it reacted sharply to the hunter movements. As a result, it changed tracks quite often. As a result, it lost time, and consequently scored lower in fitness. Its reaction to the hunters became dominant in the 3, 4, and 5 hunter case. The monolithic network lost a lot of opportunities to capture the prey even when the prey was close by (figure 6), because the dominant behaviour it learnt was to avoid the hunters.

On the other hand, the selection network was not easily perturbed by the hunters (figure 9). Furthermore, the selection network made decisions taking into account the positions of more than one agent. Note that the selection network only decides which modular network to use, and the module decides how to move. Figures 7 and 8 show how the

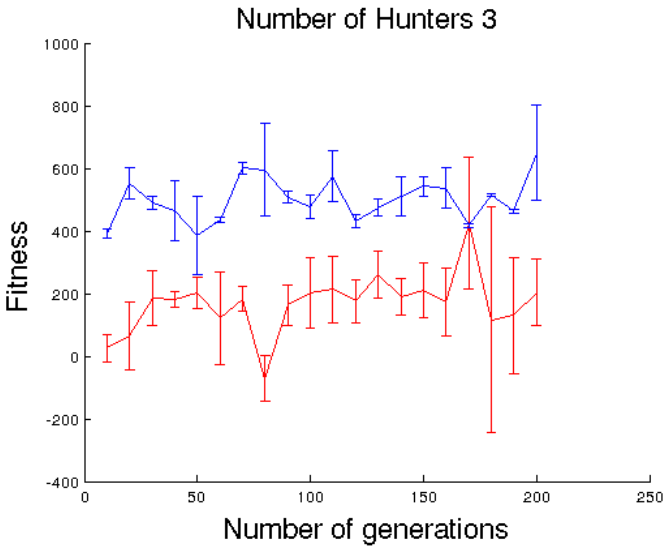


Figure 4: Champion fitness of Monolithic network (red) and Selection network (blue) with three hunters and one prey. The results have been averaged over 30 experiments. The error bars represent the standard deviation.

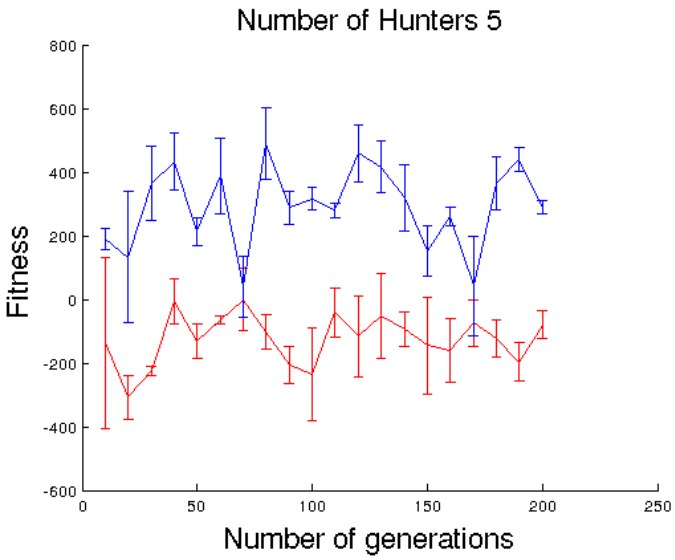


Figure 5: Champion fitness of Monolithic network (red) and Selection network (blue) with five hunters and one prey. The results have been averaged over 30 experiments. The error bars represent the standard deviation.

Number of Hidden Neurons		
Number of Hunters	Monolithic	Selection
1	136	88
2	182	96
3	120	101
4	153	107
5	163	126

Table 2: Number of hidden neurons of the champion network of generation 1000 for the monolithic network and the selection network.

predator takes into account the prey position as well as the hunter position in order to decide its next move.

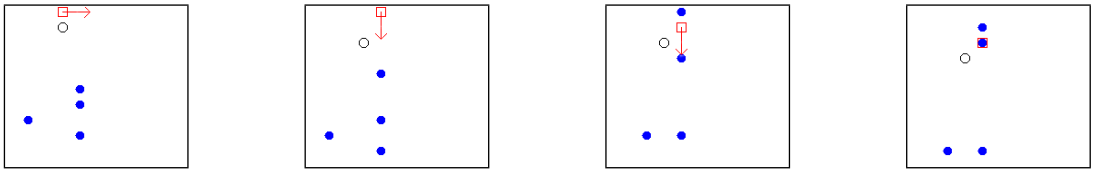
Figure 10 shows a case where the selection network chooses between the network corresponding to chasing the prey and the one corresponding to evading the hunter, depending on their positions. In figure 10(a) the predator is chasing the prey, but when the hunter gets too close, it switches to evading it as seen in figure 10(b). In the next time step, it goes back to chasing the prey as seen in figure 10(c). The selection network was observed to predominantly choose the network corresponding to chasing the prey, but occasionally selected the network corresponding to evading the hunter in case the hunter got too close while the prey was far. The selection network was also observed to choose the network corresponding to evading the hunter for the first few steps at the beginning of each episode. It should be noted that most of the time, chasing the prey also gets the predator away from the hunters, and the few times it doesn't, the predator explicitly evades the hunter.

The selection network was sometimes caught while focusing on the prey and ignoring the hunters. This is attributed to the small randomness present in the movement of the prey and hunters, as a result of which it could sometimes catch the prey while ignoring the hunters, and sometimes it was caught while exhibiting the same behavior. Overall, however, such risk taking was effective, which may be why it evolved.

Discussion and Future Work

In this paper, learning a complex task using task decomposition was shown to perform significantly better than using a monolithic network. The task decomposition performs better the more complex the domain is. Note that in the current approach task decomposition has to be done with human input. The way the task is decomposed may not be obvious or unique for most domains. Further, there may exist domains where the tasks are too tightly coupled to be amenable to task decomposition. However, when it is applicable, the results in this paper show that task decomposition is a powerful approach.

There are also multiple avenues for extensions of this work. Broadly, these can be classified as (1) changing the methods of combining subtasks, (2) changing the type of networks itself, (3) giving different types of input to the networks and (4) applying the approach to more complex domains. Apart from these four broad categories, the two other major possible extensions are co-evolving the networks and automating task decomposition. These extensions are de-

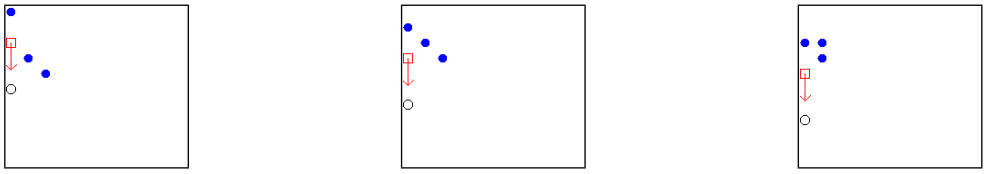


(a) (b) (c) (d)

Figure 6: Monolithic network is strongly affected by hunter movements. As can be seen in (a) the predator is very close to the prey. However in (b) it reacts sharply to the hunters especially the one at the bottom (which is closest to it in the toroidal world). As a result it loses sight of the prey, and eventually gets caught as can be seen in (d).

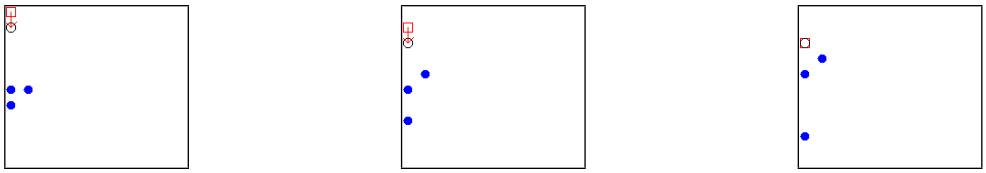
(a) (b)

Figure 7: The selection network is able to make decisions taking into account more than one agent. As can be seen in (a) the predator has agents to its right and the prey to its left. Instead of moving forward, it moves down, as shown in (b) thereby allowing it to both evade the hunters and come closer to the prey at the same time. Although it might seem that the predator only tries to minimize its distance with respect to the prey, it has to routinely avoid the hunters in order to avoid getting caught to ensure high fitness scores.



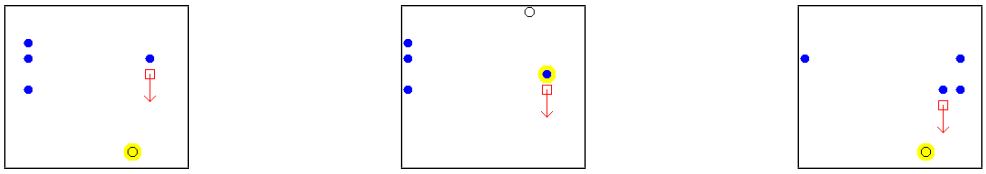
(a) (b) (c)

Figure 8: Here we see another instance where the selection network is able to make decisions taking into account more than one agent. As can be seen in (a), the predator is flanked by a hunter on the top and to its right. However, instead of moving left in order to maximize the distance from the hunters, it moves down to also simultaneously try to reduce its distance from the prey. Note that the hunters to the right of the predator move up, as in (b) to minimize the distance with respect to the previous position of the predator.



(a) (b) (c)

Figure 9: Unlike the monolithic network shown in figure 6 the selection network is not easily perturbed by hunter actions. The predator is chasing the prey in (a). As the hunters get closer in (b), the predator continues to chase the prey, unperturbed, and catches it in (c)



(a) (b) (c)

Figure 10: Illustration of switching behaviour between hunter and prey in three consecutive steps. The agent corresponding to the selected network is highlighted yellow. The predator is chasing the prey in (a) . As the hunter gets closer in (b), the predator starts evading the hunter for one time step, and goes back to chasing the prey in (c)

scribed briefly below.

In the approach described in this paper, the selection network takes inputs from all the agents in the domain and selects between the various sub-networks. Intuitively, this selection might seem limiting since the network is restricted to selecting just one of the sub-networks in each time step. To allow for more complex behaviour that is a combination of the behaviours suggested by the sub-networks, the combiner network could combine the outputs from all the sub-networks instead of selecting only one. Even using just the selection network, a hierarchy of selection networks could be developed, each one only selecting between two sub-networks. This approach would allow for a more fine-grained control of decomposition and task assignment among the networks.

Given the domain used in this paper, if the predator could keep track of the number of time steps remaining before the end of the episode, it should be able to select a better strategy. For example, if the predator knew that the time remaining is not sufficient to chase down any prey, it could concentrate on avoiding the hunters and not risk getting caught. This information could either be given to the predator as another input, or more generally, recurrent networks could be used.

Co-evolving the sub-networks and the combiner/selection network simultaneously is a promising avenue for extending this work. Rather than evolving sub-networks that only do the task optimally, a sub-network that cooperates well with the other sub-networks and the combiner network would be evolved. This approach would reduce the cases where the combiner/selection network would have to make sub-optimal choices.

There are various ways in which the domain itself might be extended for more complex tasks that might be able to take more advantage of the sub-task decomposition. For instance, it would be interesting to have teams of predators that need to cooperate to achieve the goal. Multiple subtasks that have dependencies on each other, and require a hierarchy of sub-task networks would also be an important step towards simulating complex behavior.

Developing a method to partially or completely automate the task decomposition would help reduce the human input that is required right now to specify the sub-tasks. It would also help us understand which tasks are amenable to task decomposition and how decomposition contributes to complex behavior.

Conclusion

In this paper, an approach was developed for task decomposition in the neuroevolution framework. This approach is successfully demonstrated on the predator-prey-hunter domain, an extension of the predator-prey domain where there are additonal agents (hunters) that can hunt the predators. This approach scales well as the difficulty of the task increases, and consistently performs better and more robustly than the network evolved over the whole task directly. This approach can be seen as a stepping stone to methods that discover task decomposition automatically, thus leading to development of complex general behavior in artifical agents.

References

Benda, M., Jagannathan, V., and Dodhiawala, R. (1986). On optimal cooperation of knowledge sources – an empirical investigation. Technical Report BCS–G2010–28, Boeing Advanced Technology Center, Boeing Computing Services, Seattle, WA, USA.

Floreano, D. and Urzelai, J. (2000). Evolutionary robots with on-line self-organization and behavioral fitness. *Neural Networks*, pages 431–443.

Gomez, F. and Miikkulainen, R. (1997). Incremental evolution of complex general behavior. *Adaptive Behavior*, pages 317–342.

Haynes, T. and Sen, S. (1996). *Evolving behavioral strategies in predators and prey*, volume 1042 of *Lecture Notes in Computer Science*, pages 113–126. Springer Berlin / Heidelberg.

Lee, W. (1999). Evolving complex robot behaviors. *Information Sciences*, 121(1-2):1–25.

Luke, S. and Spector, L. (1996). Evolving teamwork and coordination with genetic programming. In *Proceedings of the First Annual Conference on Genetic Programming*, pages 150–156, Cambridge, MA, USA. MIT Press.

Miller, G. and Cliff, D. (1994). *Co-evolution of pursuit and evasion I: Biological and game-theoretic foundations*. Brighton: School of Cognitive and Computing Sciences, University of Sussex.

Rajagopalan, P., Rawal, A., Miikkulainen, R., Wiseman, M. A., and Holekamp, K. E. (2011). The role of reward structure, coordination mechanism and net return in the evolution of cooperation. In *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG 2011)*, Seoul, South Korea.

Stanley, K. O., Bryant, B. D., and Miikkulainen, R. (2005). Real-time neuroevolution in the nero video game. *IEEE Transactions on Evolutionary Computation*, pages 653–668.

Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127.

Whiteson, S., Kohl, N., Miikkulainen, R., and Stone, P. (2005). Evolving soccer keepaway players through task decomposition. *Machine Learning*, 59(1-2):5–30.

Yannakakis, G. N. and Hallam, J. (2005). *AI in Computer Games: Generating Interesting Interactive Opponents by the use of Evolutionary Computation*. PhD thesis, University of Edinburgh. College of Science and Engineering. School of Informatics.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423 –1447.

Yong, C. H. and Miikkulainen, R. (2010). Coevolution of role-based cooperation in multi-agent systems. *IEEE Transactions on Autonomous Mental Development*, 1:170–186.