Nature vs Nurture: Effects of Learning on Evolution

by

Nagina Nagrani

A thesis submitted in conformity with the requirements for the degree of Masters of Applied Science
Institute for Aerospace Studies (UTIAS)
University of Toronto

© Copyright by Nagina Nagrani 2010
Abstract

In the field of Evolutionary Robotics, the design, development and application of artificial neural networks as controllers have derived their inspiration from biology. Biologists and artificial intelligence researchers are trying to understand the effects of neural network learning during the lifetime of the individuals on evolution of these individuals by qualitative and quantitative analyses. The conclusion of these analyses can help develop optimized artificial neural networks to perform any given task. The purpose of this thesis is to study the effects of learning on evolution. This has been done by applying Temporal Difference Reinforcement Learning methods to the evolution of Artificial Neural Tissue controller. The controller has been assigned the task to collect resources in a designated area in a simulated environment. The performance of the individuals is measured by the amount of resources collected. A comparison has been made between the results obtained by incorporating learning in evolution and evolution alone. The effects of learning parameters: learning rate, training period, discount rate, and policy on evolution have also been studied. It was observed that learning delays the performance of the evolving individuals over the generations. However, the non zero learning rate throughout the evolution process signifies natural selection preferring individuals possessing plasticity.
Acknowledgments

Thanks to Dr. G.M.T. D’Eleuterio, University of Toronto Institute for Aerospace Studies, for guiding my thesis efforts with his keen supervision and ideas to improve my thesis deliverables.

I would also like to acknowledge the efforts and contribution of Dr. Jekanthan Thangavelautham for providing me with the Artificial Neural Tissue controller’s code and explaining me the structure and functions of the controller. He also gave me valuable guidance for interfacing the learning methods with the controller.

Thanks to Professor Joaquim R.R.A. Martins, University of Toronto Institute for Aerospace Studies, for teaching the concepts of Genetic Algorithms in his Optimization Concepts and Applications course at University of Toronto Institute for Aerospace Studies.

Thanks to Dr. Geoffrey E. Hinton, University of Toronto, for explaining the fundamentals of machine learning during the Introduction to Machine Learning course at University of Toronto.

I thank my parents, Neelam Nagrani and Namrata Nagrani, and my brother, Neptune Nagrani, for their love, care and never ending support encouraging me throughout my graduate studies tenure.

A special thanks to Tarun Rochiramani for always being there for me and extending his unconditional technical and emotional help and support.

My humble thanks to my lab mates, Alexander Smith, Alexander Ho, Nader Abu El Samid, David Beach, Paul Grouchy, Kenneth Law, Jonathan Gammell, Peter Szabo, Francis Frenzel, Terence Fu and Kristen Facciol for maintaining a light, fun, yet educational environment at the Space Robotics Lab in University of Toronto Institute for Aerospace Studies.
## Table of Contents

Acknowledgments .......................................................................................................................... iii  
Table of Contents .......................................................................................................................... iv  
List of Tables .................................................................................................................................. vi  
List of Figures ................................................................................................................................. vii  
Chapter 1 ......................................................................................................................................... 1  
1 Introduction ..................................................................................................................................... 1  
1.1 Motivation .................................................................................................................................. 1  
1.2 Literature Review ....................................................................................................................... 3  
1.3 Background Information ............................................................................................................ 8  
1.3.1 Evolution of artificial neural networks ................................................................................. 8  
1.3.2 Temporal difference reinforcement learning ......................................................................... 10  
1.3.2.1 SARSA: On-policy TD learning ..................................................................................... 15  
1.3.2.2 Q-learning: Off-policy TD learning .............................................................................. 16  
1.3.3 ANT controller ....................................................................................................................... 16  
1.3.3.1 Structure and computation ............................................................................................. 16  
1.3.3.2 Evolution of ANT ........................................................................................................... 19  
Chapter 2 ....................................................................................................................................... 21  
2 Research Methodology ............................................................................................................... 21  
2.1 Experiment workspace ............................................................................................................. 22
2.2 Evolution ........................................................................................................................ 24

2.3 Learning .......................................................................................................................... 24

Chapter 3 ....................................................................................................................................... 29

3 Results and Discussion ........................................................................................................... 29

3.1 Case 1: Effect of type of learning on evolution ............................................................. 31

3.2 Case 2: Effect of change in environment on evolution and evolution with learning ..... 33

3.2.1 Case 2a: Effect of randomly changing environment on evolution and evolution with learning 33

3.2.2 Case 2b: Effect of one-time change in environment on evolution and evolution with learning 35

3.3 Case 3: Effect of training period on evolution ............................................................... 37

3.4 Case 4: Effect of discount rate on evolution ................................................................. 39

3.5 Case 5: Effect of learning policy on evolution ............................................................... 41

Chapter 4 ....................................................................................................................................... 43

4 Conclusion and future research work ................................................................................... 43

References ..................................................................................................................................... 45
List of Tables

Table 1: Sensory inputs for the task [Thangavelautham et al. 2007]........................................... 23

Table 2: Behaviors to execute the task [Thangavelautham et al., 2007]. ...................................... 24

Table 3: Summary of parameters used for the five cases. ............................................................ 25
List of Figures

Figure 2: Fitness surface with and without learning. Redrawn from Nolfi and Floreano, 2000. .................4

Figure 3: Evolutionary surface and Learning surface for fitness and performance of individuals ‘a’ and ‘b’ for different combination of weights [Nolfi and Floreano, 2000]. ...........................................5

Figure 4: a) Best and average fitness, b) average learning rate from Mayley’s experiment results [Mayley, 1996].................................................................................................................................6

Figure 1: Example of the workspace used for the resource gathering task simulation.........................7

Figure 5: a) Biological illustration of neurotransmission. b) Artificial neural network connection9

Figure 6: One-point crossover at third position [Boukerche, 2006, 373]. ..............................................10

Figure 7: Reinforcement learning method: interaction between agent (individual) and environment [Sutton and Barto, 1998]........................................................................................................11

Figure 8: Diagram of brain: Left diagram shows a section of the frontal lobe and the striatum. Right diagram shows the substantia nigra and striatum: caudate nucleus and putamen. [Conn, 2005, 8]. ........................................................................................................................................14

Figure 9: Arrangement of motor neurons and their synaptic connections [Thangavelautham et al., 2007]. ........................................................................................................................................17

Figure 10: Coarse-coding: activation/inhibition of neural network by decision neurons [Thangavelautham et al., 2007]. ........................................................................................................................................18

Figure 11: Gene map for motor neuron, decision neuron and tissue gene [Thangavelautham et al., 2007]. ........................................................................................................................................19

Figure 12: Sensory inputs: a) resource detection, b) workspace color template detection, and obstacle detection [Thangavelautham et al., 2007]........................................................................................................23

Figure 13: Board after one-time change in environment........................................................................28
Figure 14: Learning rate for different environment conditions .................................................... 30

Figure 15: Maximum fitness for evolution vs evolution + SARSA vs evolution + Q-learning (Case 1) .......................................................................................................................................................... 32

Figure 16: Average fitness for evolution vs evolution + Q-learning vs evolution + SARSA (Case 1) ................................................................................................................................................... 32

Figure 17: Maximum fitness for evolution vs evolution + learning in randomly changing environment (Case 2a) ......................................................................................................................... 34

Figure 18: Average fitness for evolution vs evolution + learning in randomly changing environment (Case 2a) ......................................................................................................................... 34

Figure 19: Maximum fitness for evolution vs evolution + learning in one-time changing environment (Case 2b) .......................................................................................................................... 36

Figure 20: Average fitness for evolution vs evolution + learning in a one-time changing environment (Case 2b) .......................................................................................................................... 36

Figure 21: Maximum fitness for evolution vs evolution + learning for different training periods (Case 3) .................................................................................................................................................. 37

Figure 22: Average fitness for evolution vs evolution + learning for different training periods (Case 3) .................................................................................................................................................. 38

Figure 23: Polynomial fitting example to explain poor fitting and over-fitting [Bishop, 2006] .. 39

Figure 24: Maximum fitness for evolution vs evolution + learning for different discount rates (Case 4) ................................................................................................................................................... 40

Figure 25: Average fitness for evolution vs evolution + learning for different discount rates (Case 4) ................................................................................................................................................... 40

Figure 26: Maximum fitness for evolution vs evolution + learning for different ε value in ε-greedy policy (Case 5) .......................................................................................................................................... 41
Figure 27: Average fitness for evolution vs evolution + learning for different $\varepsilon$ value in $\varepsilon$-greedy policy (Case 5) ................................................................................................................................................................. 42

Figure 28: Maximum fitness values for Case 1, 3, 4 and 5 vs evolution .............................................. 43
Chapter 1

1 Introduction

The following thesis is a study to understand the effects of learning on evolution of artificial neural networks in order to optimize neural network controllers for robotic applications. The choice of artificial neural networks as controllers have derived its inspiration from the nervous system found in multicellular organisms. To perform a task these organisms either genetically inherit the ability from their parents as they evolve over generations or they learn to perform tasks during their lifetime. The ability to learn has given rise to questions in biology as well as artificial intelligence studies such as does learning chalk an evolutionary pattern different from the one that exclusively relies on evolution? Does the incorporation of learning produce more optimized individuals? If yes, will these individuals appear in earlier generations as compared to individuals that evolve without learning? Also, will an individual that possesses the ability to learn survive changing environments? While biologists try to answer these questions using experiments such as observing operating cost of learning in Drosophila melanogaster [Mery and Kawecki, 2004] and predicting reward signal of dopamine neurons [Shultz, 1998], artificial intelligence researchers are analyzing these effects of learning on evolution using simulations such as Hinton and Nowlan’s [1987] computational simulation to find the a specific set of 20 connections of 1’s and 0’s in a neural network using a simple genetic algorithm and learning method, and as Mayley [1996] compares genetic assimilation [Waddington, 1942] in two evolutionary models, first a NK Fitness model that uses stochastically generated fitness landscapes from N genes and K interactions between genes [Altenberg, 1996], and a second model that uses the evolution of a controller for a real world robot.

1.1 Motivation

One of the most aspiring researches in robotics has been to design controllers to improve autonomy of systems that should have the capability to perform tasks with little or no human intervention and respond to any change in the environment in order to adapt and survive while executing the task assigned to the system. For example, the rovers designed for the NASA’s Mars exploration missions are semi-autonomously controlled. Due to communication delays to send signals from Earth to Mars (and vice versa), at the beginning of a Martian day the ground
control on Earth sends a sequence of commands to the rovers such as traversing a set of waypoints or drilling or collecting samples, the rovers execute these commands autonomously (based on their designed control system) during the day and before it shuts down at night it sends collected data back to ground control based on which the ground control plans the following day’s events to perform a high level task [Carsten et al., 2007]. The commands sent to the rovers are a result of decomposition of the high level task. It depends on the control system designer how well the task has been decomposed to design an optimum solution. However, if the designer has limited information about the environment where the system will perform its task, the system needs to possess its own intelligence to deal with unforeseen circumstances. This will also minimize human intervention during task execution and save time for mission accomplishment.

Another option for developing controllers that can autonomously survive in a dynamic environment that we have incomplete information about has been derived from biology: Evolution. Darwin’s famous Theory of Evolution was one of the solutions for the above mentioned requirement. Darwin’s Theory of Evolution states that all living forms of organisms have a common ancestor that had a simpler structure. Over generations these simpler forms of life underwent mutations. The mutations that were advantageous for the survival of the organism persisted in that specie. On the other hand, those species that were not benefited by the mutations for their survival slowly became extinct. This is in fact a more complex process. The genetic material inherited by individuals from their parent(s), either sexually or asexually, undergoes small mutations before the individual is born. The genetic information, called the genotype, thus obtained is mapped to the individual’s phenotypic information. That is, the genotype is encoded with information about the individual’s body. This is a one way mapping from genotype to phenotype [Weismann, 1893]. On the contrary, the phenotype does not directly influence the genotype. However, the performance of the phenotype in the contemporary environment results in the survival of the individual’s genotype in the population, which is subsequently passed on to the following generations, thus indirectly affecting the existence of the genotype. This process is called natural selection. These responses of the phenotype to the environment and mutations together lead to new species over many generations. After many generations when a particular trait survives in the genotype of most individuals in the population, this results in genetic assimilation of that trait [Waddington, 1942]. The classic example of peppered moth in United Kingdom helps in understanding this theory. Peppered moth existed in United Kingdom in light
and dark colors. However, during the industrial revolution, due to the soot deposits on the tree trunks, dark peppered moths could camouflage themselves from their predators. This led to an increased reproduction rate in dark moths leading to their majority in a few generations.

Similarly, the ability to learn to survive or adapt in an environment is also a phenotypic characteristic in individuals that affects evolution. For example, one of the precursors to the evolution of man from apes was to learn to hunt for food and defend himself from predators. The weak physiological form and size of apes as compared to its stronger predators required the ancestors of human beings to design tools for self defense and hunting. Genetic mutations for larger brain size, \(600-800 \text{ cm}^3\), lead to these intelligent species [Foley, 1997]. With the capability of the evolved brain size they learned to design tools using their fore limbs that in later generations became smaller than their hind limbs and developed an opposing thumb for enhancing their dexterity as seen in the fossils of Homo habilis, the skillful ancestors of humans [Tomkins, 1998]. Also, as per the archaeological studies Homo habilis originated from East Africa, these species needed to cool their larger brains to save themselves from heat stress. Thus, their head needed to be held higher with a cranial blood vessel network that worked like a car radiator. Busy fore limbs and a need to keep the brain cooler resulted in bipedalism, thus evolving the structure of bones in the limbs, hips and shoulders [Tomkins, 1998]. Thus, in order to survive in a challenging environment, learning to design tools and stand on their hind limbs was required. The traits acquired by learning were later assimilated in the genotypes of Homo sapiens (humans). Thus, we see that in the biological life, evolution of a population and learning by an individual go hand in hand for evolution of life forms.

Inspired by the evolution of biological life forms, it was studied that evolution of controllers in artificial life can allow a system to autonomously adapt to a dynamically changing environment. However, this triggered the question whether genetic evolution alone will be more beneficial for these controllers or learning can accelerate and guide evolution of artificial controllers in an attempt to find an optimum controller.

1.2 Literature Review

Among the earlier artificial intelligence researchers, Baldwin [1896] and Waddington [1942] argued that learning can significantly affect evolution. However, the effect of learning in an individual’s lifetime on evolution in the following generations is time consuming. As evolution
favors individuals with higher fitness, it will select individuals that already have the necessary features for survival, which would otherwise be learned. Thus, Baldwin suggested that although traits acquired by individuals from learning resulting in increased fitness can spread to the next generation, the evolutionary path tends to search for innate traits that can replace learned traits to overcome cost of learning (time consumption). Hinton and Nowlan [1987] computationally modeled how learning can “help and guide” evolution of neural networks. They argued that “A neural network confers added reproductive fitness on an organism only if it is connected in exactly the right way. In this worst case, there is no reasonable path towards the good net and a pure evolutionary search can only discover which of the potential connections should be present by trying possibilities at random. The good net is like a needle in a haystack.” This can be explained by the following graphical representation of Hinton and Nowlan’s model.

Figure 2: Fitness surface with and without learning. Redrawn from Nolfi and Floreano, 2000.

Their experiment involved individuals with genotypes having 20 genes, each with a value 0 or 1. The task was to search for a unique combination of these genes that resulted in high fitness value. The solid line in Figure 2 represents fitness of an individual with a combination of genes obtained without learning. Finding a particular combination of genes that will give a high fitness value for an individual while searching randomly will be a tedious task for evolution. The learning method involved genes having three alternative values, 0, 1 and a modifiable gene ‘?’.

This modifiable gene was learned to have values 0 or 1 in order to find the desired combination of genes. The dashed line in Figure 2 represents fitness obtained after learning. This curve suggests that individuals at suboptimal fitness levels can acquire higher fitness by learning. Thus,
as per the Baldwin’s effect, if these learned traits persist in the following generations, they can be incorporated in the genotype of the future generations.

However, this model has some shortcomings [Nolfi and Floreano, 2000].

1) Learning is modeled as a random task.
2) A static environment is assumed for this model.
3) The learning space and evolutionary space are completely correlated.

Nolfi and Floreano [2000] explain the effect of learning on evolution in a correlated and uncorrelated environment.

![Figure 3: Evolutionary surface and Learning surface for fitness and performance of individuals ‘a’ and ‘b’ for different combination of weights [Nolfi and Floreano, 2000].](image)

In Figure 3, we see that individuals ‘a’ and ‘b’ have the same fitness at birth. The movement of ‘a’ and ‘b’ during evolution is in the direction that increases the performance of combination of weights. Individual ‘a’ shows an increase in fitness on the evolutionary surface with an increase in performance on the learning surface, whereas individual ‘b’ shows a decrease in fitness on the evolutionary surface for an increase in performance on the learning surface. Individual ‘a’ is in a
region where the evolutionary and learning surfaces are completely correlated, whereas individual ‘b’ is in a region where evolutionary and learning surfaces are completely uncorrelated. Although, ‘a’ and ‘b’ had the same fitness at birth, ‘a’ has a higher probability to be selected by evolution than ‘b’ does, as ‘a’ possesses the innate capability to acquire higher fitness and will display good performance with learning during its lifetime.

Moreover, learning incurs some significant costs. Mayley [1996] lists these costs as time-wasting, incorrect behavior, genotypic complexity and waste of energy in performing a set of behaviors before selecting the most appropriate behavior. Also, he demonstrates that due to genetic assimilation, learning will eventually disappear from the population. See Figure 4.

![Figure 4: a) Best and average fitness, b) average learning rate from Mayley’s experiment results](Mayley, 1996).

On the other hand, the benefits of learning may outweigh its costs in a fast changing environment where the individual has to adapt to the dynamic environment in order to maintain or improve its fitness for survival. Also, if Mayley argues that eventually learning will disappear from the population, this should happen when the population is successful at accomplishing its evolutionary tasks, else it will have to continue learning to fulfill the evolutionary tasks assigned. In an extreme worst case scenario, if the environment changes and the individual’s genotype does not have the traits to deal with the change, the individual will have to learn to adapt itself in
the changing environment. However, if the individuals have the inherited ability to learn these traits, then evolution would favor them.

Thus, a tradeoff between costs and benefits of learning in evolution needs to be established. This thesis will weigh the costs and benefits of learning in evolution and map the same for the five cases considered for this study.

In this thesis the effects of learning on evolution will be studied using biologically inspired algorithms in a simulated environment. The task used to demonstrate this is to collect scattered resources in a designated area. The simulation involves a two dimensional grid (confined by a blue border) with a rectangular designated area (red-orange border), scattered resources (represented by brown squares) and three agents (represented by green circle occupying 4 squares each) using an evolved Artificial Neural Tissue (ANT) controller [Thangavelautham et al., 2007] to perform the resource gathering task. See Figure 1.

Figure 1: Example of the workspace used for the resource gathering task simulation.

The agents have been evolved for 500 generations. The learning method used is the Temporal Difference Reinforcement Learning [Sutton and Barto, 1998]. Five cases have been planned to
compare the agents’ performances: effect of type of learning, environment, training period and other learning parameters specific to the learning method chosen.

The following section provides background information about the fundamentals of evolution of artificial neural networks, temporal difference reinforcement learning, and ANT controller.

1.3 Background Information

This section of the chapter provides basics of artificial evolution of neural networks, temporal difference reinforcement learning, and ANT controller. The research methodology described in the following chapter uses the above mentioned expertise in the field of artificial life.

1.3.1 Evolution of artificial neural networks

As understood in biology, a neural network is a collection of neurons with gaps between neurons called synaptic gaps. These neurons transmit electrochemical signals via these synaptic clefts. These electrochemical signals are either obtained from the environment as sensory inputs and transmitted to a decision making centre, or these signals are transmitted from the decision making centre to action/motor neurons in order to perform an action as a response to the sensory input. When a signal arrives at a neuron, the neuron transmits some chemicals called neurotransmitters that influence the next neuron (postsynaptic neuron) in an excitatory or inhibitory way. If the postsynaptic neuron is more excited than inhibited, the signal moves ahead to that neuron, thus establishing a path for signal transmission in order to sense or react to the environment. Similarly, in artificial neural networks, signal transmission is performed using weights, w_k^i, from node k to node i (neurons), that mimic the effects of neurotransmitters in the synaptic clefts. See Figure 5 for a diagrammatic illustration of the biological inspiration derived by artificial neural networks.
The objective of artificial intelligence researchers is to find the set of synaptic weights that result in the optimum controller. Evolution is one technique to optimize controllers by searching for the right set of weights. A neural network controller comprising of a particular set of weights is called an individual. These weights are stored in the form of strings termed as genes. This genetic information is transferred from one generation (iteration) to another by using genetic operators such as crossover, mutation and selection of individuals. Similar to biological evolution, survival of the fittest rule applies here as well. The performance of an individual is
referred to as the fitness of the individual. The optimization tool that includes these operations is called a Genetic Algorithm (GA).

A simple genetic algorithm tracks the following steps:

1. Start with a population of individuals with randomly generated genes.
2. Evaluate the fitness of each individual.
3. Determine the mating pool either by selecting individuals at random or allowing individuals with better fitness values to reproduce more often than others.
4. Crossover operation where genetic information between two parents are swapped at a point in the gene to generate offsprings. See Figure 6.

![Parents and Offspring](image)

Figure 6: One-point crossover at third position [Boukerche, 2006, 373].

5. Mutation operation where randomness is added to the offspring’s genes for diversity. This is done by changing a weight value at a given mutation rate.
6. Evaluate offspring’s fitness.
7. Selection operation where worse individuals are replaced by better offspring.
8. Return to Step 3.
9. At the end of predetermined number of generations (iterations), identify the best individual.

1.3.2 Temporal difference reinforcement learning

Machine learning is another way to optimize a neural network controller by updating weights using errors obtained during training phase. As concluded by Hinton and Nowlan’s [1987] experiment, learning gives individuals with lower innate fitness (at birth) an opportunity to improve their fitness during their lifetime. In order to verify this statement for evolutionary robotics controller mimicking biological systems in a dynamic environment, it is important to use a biologically inspired learning method. This can also help us understand the influence of
learning on the evolution of biological forms. A learning method maps sensory information to output behavior or action. There are three types of learning method: supervised learning, unsupervised learning and reinforcement learning. In supervised learning, the designer of the controller knows the final solution of a task, i.e. the correlation between sensory inputs and output behaviors are known, and the weights are learned by using the difference between the final solution and the performance of the controller with the current set of weights. This is also called offline learning. However, if we need to build an autonomous controller, there are certain unforeseen environmental circumstances that the designer cannot train the controller for. Thus, an unsupervised learning method can be considered to build such controllers. In an unsupervised learning method, there is no supervisory signal from either the designer or the environment. Instead, an individual needs to learn the association between sensory cues. However, establishing this association leads to a noisy feedback [Littman, 1995, pp. 470-472]. Also, this could be a more time consuming learning method that may lead to learning incorrect behaviors [Mayley, 1996]. The third kind of learning, reinforcement learning, is less supervisory in nature. Instead of training individual for a particular action to be taken for given sensory information, reinforcement learning allows an individual to acquire a feedback signal (reward) from the environment about how good or bad the previous action was for the sensory information obtained at that time, and thus update weights. See Figure 6 for a diagrammatic description of the reinforcement learning method.

![Figure 6: Diagram of reinforcement learning method](image)

**Figure 7**: Reinforcement learning method: interaction between agent (individual) and environment [Sutton and Barto, 1998].
In Figure 7, at time $t$ the sensory inputs provide state $s_t$, the environment provides reward $r_t$, which is a feedback due to the previous action, resulting in action $a_t$ by the agent, due to which the agent receives a reward $r_{t+1}$ from the environment. This reward and the sensory inputs at time $t+1$ help in evaluating the next action. Over the training phase this allows the controller to map a relationship between sensory inputs and actions. The objective in this learning method is to maximize the total reward received from the environment, which is quantified by a value function $V(s)$. Thus, $V(s)$ helps in mapping a state to the action to be taken in that state. This mapping is termed as policy, $\pi$. This map is then used to collect the reward for a state action pair in order to update the value function. This continues till the policy is optimized.

The different types of reinforcement learning methods are: Dynamic Programming, Monte Carlo method and Temporal-Difference (TD) learning.

Dynamic Programming (DP) requires complete information about the environment in order to estimate $V(s)$. This method is governed by bootstrapping, that is $V(s)$ is estimated based on the estimation of $V(s')$, value of the next state $s'$. For each state $s$, $V(s)$ is compared to its previous value till convergence occurs, thus increasing the computation time for larger state sets. Consequently, the policy is updated as the state-action pairing, which for an action results in maximum $V(s)$. The update continues till the state-action pairing does not change. However, this policy update rule does not allow exploration for better actions for a given state.

Monte Carlo method does not assume complete knowledge of the environment. Instead, it is based on learning from experience. For this method, a task’s timeline is divided into episodes and $V(s)$ and $\pi$ are estimated at the end of the episode. The $V(s)$ estimate is a very crude one, which is equal to the average of the rewards (on-policy method: using and updating the same policy) or a weighted average of the rewards (off-policy method: using one policy while updating another) collected in that episode. Monte Carlo method has an edge over DP method as it does not require complete knowledge of the environment, instead interacts with the environment in order to optimize its policy. Also, they can be used and are more efficient when given a simulated sample episode. Furthermore, they require less computation time as they focus on episodes rather than step by step updates of $V(s)$ and $\pi$. However, their convergence rule has not yet been fully established, hence this method has been least tested. Also, one needs to wait till the end of an episode to know the rewards obtained by the state-action pairs.
TD learning method learns a policy based on experience as well as timely learned estimates for $V(s)$. The advantage that TD learning has over DP learning is that it does not require any information about the environment before starting the optimizing process. Also, the utilization of timely $V(s)$ updates proves to be advantageous for TD learning method over Monte Carlo method for applications that require long episodes. Sutton and Barto [1998] guarantee convergence to the correct state-action values by keeping the policy fixed. Also, Sutton and Barto claim that TD learning method learns faster than DP and Monte Carlo learning method. Furthermore, as explained later, TD learning method is biologically inspired.

ANT being a biologically inspired controller, in this thesis mimicking biological systems as closely as possible will help better understand the effects of learning on evolution for the sake of biology as well as artificial intelligence, hence the TD learning method is considered to be used during the lifetime of an individual (See Chapter 2).

TD learning method is highly inspired by the functions of a mammalian brain. A neurotransmitter called dopamine acts as the rewarding agent in a mammalian brain. The dopamine neurons are present in the midbrain (substantia nigra), that release dopamine to striatum based on a stimulation received from the sensory cortex assessment of sensory inputs. The striatum contains the caudate nucleus (more cognitive) and putamen (more motor), that are made of two components: striosome and matriosome. The striosome acts as the reward predictor which uses the concentration of dopamine to determine the difference between the expected and received rewards. The matriosome uses this information for action selection that is transmitted to the frontal lobe (motor cortex) to perform the decided action. The density of striosome is more in the caudate nucleus and the density of matriosome is more in the putamen [Rivest et al., 2004]. See Figure 8 to identify the different areas of the brain mentioned above.
The following algorithm from chapter 6.1, Sutton and Barto’s Reinforcement Learning: An Introduction (1998) helps understand the similarity between the above mentioned understanding of how the brain learns, and TD learning method.

1) Initialize value function, $V(s)$ arbitrarily, and $\pi$, the policy. For TD learning policy is fixed as the rate at which the individual tries a random action in order to make sure the individual does not get stuck at a local optimum. This allows the controller to strike a balance between exploitation and exploration of optimum actions.

2) Repeat for each episode:
   a. Initialize state $s$. In the biological system this is the sensory information obtained from the sensory cortex.
   b. Repeat for each step of the episode:
      i. Decide action $a$ for which $V(s)$ is maximum for state $s$. Or consider a random action $a$ at the rate of $\pi$. In the brain, this decision is made by the matriosome in the striatum.
      ii. Take action $a$ (this information is transferred by the striatum to the frontal lobe, i.e. motor cortex), observe reward $r$ (dopamine concentration stimulated by sensory cortex), and next state $s'$ (information obtained from the sensory cortex).
iii. Calculate $V(s) \leftarrow V(s) + \alpha [r + \gamma V(s') - V(s)]$, where $\alpha$ is the learning rate and $\gamma$ is the discount the algorithm gives to the next state’s value in updating the value for the current state. This evaluation is done by the striosome in the striatum.

iv. $s \leftarrow s'$.

c. Until $s$ is terminal.

TD learning method is further divided into two classes: on-policy TD learning called SARSA (State Action Reward State Action), and off-policy TD learning called Q-learning. These two algorithms aim to find the optimum estimate for action-state value function $Q(s, a)$, a value that represents the reward of performing action $a$ in state $s$. These two classes differ based on how the policy is used. There are three common policies:

1) $\varepsilon$-greedy – the controller selects a random action uniformly, independent of $Q(s, a)$, with a small probability, $\varepsilon$.
2) $\varepsilon$-soft – the controller selects an action holding the maximum $Q(s, a)$ with a probability $1-\varepsilon$. Rest of the time it uniformly chooses an action at random.
3) Softmax – this method allows the controller to try and ignore the worst actions. Unlike $\varepsilon$-greedy and $\varepsilon$-soft, this method assigns a weight to the actions based on their $Q(s, a)$. An action is selected with a probability $\pi = \frac{e^{Q(s,a)/\tau}}{\sum_a e^{Q(s,a)/\tau}}$, where $\tau$ is the temperature.

1.3.2.1 SARSA: On-policy TD learning

On-policy TD learning implies that $Q(s, a)$ is estimated based on policy $\pi$ for all states $s$ and actions $a$. The algorithm as obtained from Sutton and Barto [1998] is as follows:

1. Initialize $Q(s, a)$ arbitrarily.
2. Repeat for each episode:
   a. Initialize $s$.
   b. Choose $a$ from $s$ using policy derived from $Q$ (e.g. $\varepsilon$-greedy).
   c. Repeat for each step of the episode:
      i. Take action $a$, observe $r, s'$.
      ii. Choose $a'$ from $s'$ using policy derived from $Q$ (e.g. $\varepsilon$-greedy).
      iii. $Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma Q(s', a') - Q(s, a)]$.
      iv. $s \leftarrow s', a \leftarrow a'$.
   d. Until $s$ is terminal.
1.3.2.2 Q-learning: Off-policy TD learning

Off-policy TD learning implies that the optimum $Q(s, a)$ is estimated directly from the best $Q(s', a')$ for state $s'$. This estimation is done independent of the policy. The algorithm as obtained from Sutton and Barto [1998] is as follows:

1. Initialize $Q(s, a)$ arbitrarily.
2. Repeat for each episode:
   a. Initialize $s$.
   b. Repeat for each step of the episode:
      i. Choose $a$ from $s$ using policy derived from $Q$ (e.g. $\epsilon$-greedy).
      ii. Take action $a$, observe $r, s'$.
      iii. $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'}Q(s', a') - Q(s, a)]$.
      iv. $s \leftarrow s'$.
   c. Until $s$ is terminal.

1.3.3 ANT controller

The Artificial Neural Tissue (ANT) controller is highly inspired by the functioning of a neuron. A neuron not only communicates through electrical signals as explained in section 1.3.1, but also chemically which selectively activates neurons. The genome of the ANT controller, designed by Thangavelautham et al. [2007], is evolved in an artificial Darwinian fashion. ANT controller combines a typical feedforward neural network structure with coarse-coding [Albus, 1981].

1.3.3.1 Structure and computation

ANT is a three dimensional neural network comprising of decision neurons and motor neurons. As seen in Figure 9, motor neurons are arranged as spheres in a regular rectangular lattice structure. Each neuron $N_\lambda$ occupies position $\lambda = (l, m, n) \in \mathbb{Z}^3$. The state of the neuron is $s_\lambda$, which is binary in nature. Each $N_\lambda$ neurons receive nominal inputs via synaptic connections from 3 x 3 neurons from the lower layer, $N_\kappa$, occupying positions $\kappa = \pi(\lambda) = \{(i, j, k) | i = l - 1, l, l + 1; j = m - 1, m, m + 1; k = n - 1\}$. 
Figure 9: Arrangement of motor neurons and their synaptic connections [Thangavelautham et al., 2007].

Each neuron is activated by threshold functions that depend on the weighted input $\sigma$:

$$
\psi_{down}(\sigma) = \begin{cases} 
0, & \text{if } \sigma \geq \theta_1 \\
1, & \text{otherwise}
\end{cases}
$$

$$
\psi_{up}(\sigma) = \begin{cases} 
0, & \text{if } \sigma \leq \theta_2 \\
1, & \text{otherwise}
\end{cases}
$$

$$
\psi_{ditch}(\sigma) = \begin{cases} 
0, & \min(\theta_1, \theta_2) \leq \sigma < \max(\theta_1, \theta_2) \\
1, & \text{otherwise}
\end{cases}
$$

$$
\psi_{mound}(\sigma) = \begin{cases} 
0, & \sigma \leq \min(\theta_1, \theta_2) \text{ or } \sigma > \max(\theta_1, \theta_2) \\
1, & \text{otherwise}
\end{cases}
$$

(1.1)

The weighted input to neuron $N_{\lambda}$ is given as

$$
\sigma_{\lambda} = \frac{\sum_{k \in \Omega(\lambda)} w^k_{\lambda} s_k}{\sum_{k \in \Omega(\lambda)} s_k}
$$

(1.2)

Where, $(\theta_1, \theta_2) \in \mathbb{R}^2$ are threshold parameters, $w^k_{\lambda} \in \mathbb{R}$ is the weight of the synaptic connection from $N_k$ to $N_{\lambda}$, and $s_k$ is the state of $N_k$. Threshold functions listed in (1.1) can be summarized as

$$
\psi = (1 - k_1)[(1 - k_2)\psi_{down} + k_2\psi_{up}] + k_1[(1 - k_2)\psi_{ditch} + k_2\psi_{mound}]
$$

(1.3)
Where, $k_1$ and $k_2$ are binary, i.e. they take on values 0 or 1. Thus, $\psi$ is encoded in the genome as $k_1, k_2$, and threshold parameters $\theta_1$ and $\theta_2$. See gene map in Figure 11. This is the structure of the feedforward neural network used by ANT.

As explained earlier, ANT uses a combination of the feedforward neural network and coarse coding. The latter is introduced in ANT as decision neurons. In Figure 10 decision neurons are shown as small rectangular boxes at nodes in the lattice structure shown in Figure 9. The location of these decision neurons is determined by evolution.

Figure 10: Coarse-coding: activation/inhibition of neural network by decision neurons [Thangavelautham et al., 2007].

As understood from biology, certain neurons also communicate chemically. These neurons are termed as decision neurons. A decision neuron $T_\mu$ has a state $\mu$, which is binary in nature. $T_\mu$ receive signals from all input sensory neurons $N_\alpha$. These signals are given by

$$s_\mu = \sum_{\alpha} v_\mu^\alpha s_\alpha$$

The weighted input required by the decision neurons is given by

$$s_\mu = \sum_{\alpha} v_\mu^\alpha s_\alpha$$

Where, $v_\mu^\alpha$ is the weight from input $N_\alpha$ to $N_\mu$. If $s_\mu = 0$, the decision neuron is inactive. However, if $s_\mu = 1$, the decision neuron emits a neurotransmitter with concentration $c_\mu$, that influences motor neurons in its vicinity, as shown by the concentration fields in Figure 10. The concentration fields are shown as rectangular boxes extending by $\pm d_\mu^r$, where $r = 1, 2, 3$ indicating...
three perpendicular directions from the decision neuron. These three dimensions, \( \mu \), \( d \) and concentration \( c_a \) are encoded in the decision neuron genome. See gene map in Figure 11. Motor neurons that lie in the overlapping highest concentration fields get activated, while others get inhibited. As a motor neuron is excited, it transmits signals as shown in Figure 9 and equation 1.2. Only those motor neurons activated in this manner establish the neural network from sensory neurons to the output neurons. Output neurons, \( N_{\omega j} \), \( j = 1\ldots n \), with state \( \Omega = \{s_{\omega 1}, s_{\omega 2}, \ldots, s_{\omega j}, \ldots, s_{\omega n}\} \), where \( b \) is the number of behaviors (actions). Each output neuron is dedicated to a behavior. If \( s_{\omega j} = 1 \), the output neuron \( N_{\omega j} \) votes for behavior \( k \), whereas if \( s_{\omega j} = 0 \), the output neuron does not vote. Since multiple neural networks can have an output neuron that vouches for the same behavior, the controller is made deterministic in nature. It calculates 
\[
p(k) = \sum_{s_{\omega j}=1}^{n_b} \frac{s_{\omega j}^k}{n_k},
\]
where \( n_k \) is the number of networks that vouch for output behavior \( k \). If \( p(k) \geq 0.5 \), behavior \( k \) is activated.

Figure 11: Gene map for motor neuron, decision neuron and tissue gene [Thangavelautham et al., 2007].

### 1.3.3.2 Evolution of ANT

Gene map in Figure 11 shows that each gene has a specifier \( D \) that describes the gene’s function. An autonomous program (constructor protein in biology) reads this specifier and translates the genetic information into a class cell (cell descriptor protein). The gene activation parameter states whether the descriptor protein should be created or not. If gene activation parameter is 1, the program grows the cell into a tissue by replication in which each cell is located relative to a specified seed-parent address. The cell death parameter in motor control neuron gene indicates if the cell commits suicide or not. If the cell commits suicide it continues to occupy space in the
tissue, however remains dormant. Evolution decides if the gene information of this cell should be used in the next generation or not by changing this bit.

In the tissue gene, the neuron replication probability is used for cell replication, where the cell (parent cell) having the highest replication probability replicates more often than the rest of the cells in the tissue. During replication, m% (m is a selected from a uniform random distribution) of the parent cell information is transferred to the daughter cell, and the rest of the information is initialized by a uniform random distribution. This helps provide an opportunity to incorporate diversity in the tissue. The neuron replication ratio helps determine the type of neuron born. The new cell occupies a space beside the parent cell positioned either top, bottom, north, south, east or west of the parent cell.

In order to explain the details of this study, this document is divided into four chapters. This chapter provided motivation, literature review, and background information about evolution of artificial neural networks, temporal difference reinforcement learning, and ANT controller. Chapter 2 describes the research methodology. Chapter 3 discusses the observations and analysis of the simulation results for the five cases listed above followed by Chapter 4 that concludes this study by providing its contribution to understand the effects of learning on evolution and some recommended future research work.
Chapter 2

2 Research Methodology

In order to evaluate the effects of learning on evolution, simulations have been conducted using the resource gathering task described by Thangavelautham et al. (2007) to demonstrate the performance of ANT. There are five cases for which simulations are performed:

Case 1: Effect of type of learning method used on evolution

The two types of learning methods used are SARSA and Q-Learning. Both these methods are described in Sections 1.3.2.1 and 1.3.2.2 respectively. The type of learning method provides a learning rule that may affect the weight updates and hence steer the evolution path in a different direction or pace.

Case 2: Effect of change in environment on both evolution with learning and pure evolution.

The dynamics of the environment provide a major threat to many control system designers as these may result in encountering unforeseen circumstances that the controller may not be designed for. Also, environment provides a feedback element to the reinforcement learning method, thus becoming an important factor to consider while assessing the effects of learning on evolution. Two sub cases have been used to study this effect.

Case 2a: Random change in environment for every generation.

The random change in environment for this simulation is produced by randomly changing the location, size and shape of the designated area for resource collection for each generation.

Case 2b: One-time change in environment after 250 generations.

Half way through the number of generations considered for this thesis study the location, size and shape of the designated area is changed as shown in Figure 13.

Case 3: Effect of training period on evolution.
Training period of a learning method decides the opportunity given to an individual to learn as many input-output network connections as it can, thus preparing it to apply what it learned in order to improve its fitness. However, shorter training period may not allow it to learn as much, whereas longer training period will lead to increase in noise in network connections leading to confusion in the individual, thus increasing the chances of learning wrong behaviors.

Case 4: Effect of discount rate on evolution.

Discount rate allows the individual to consider a percentage of the possible resulting states for an action taken such that it can accordingly calculate the accuracy of the decided action. The consideration of the possible results of an action can affect the performance of learning and thus influence the path of evolution.

Case 5: Effect of learning policy.

As seen earlier, the learning policy for TD learning method is fixed in order to have a quicker convergence. However, an optimum policy is needed for formatting a fine mapping from the sensory inputs to the output behaviors. The $\epsilon$-greedy method is used for the policy as it has been successfully tested for various applications [Sutton & Barto, 1998].

For each case 5 runs are executed. This chapter describes the research methodology followed to demonstrate the above. This has been explained in three sections: Experiment workspace, evolution, learning.

### 2.1 Experiment workspace

In this task a workspace bounded by a restrictive border shown in blue is scattered with resources, which need to be collected in a designated area shown as the red-orange border. See Figure 1. The designated area is confined by two colors to indicate if an agent is inside (orange border) or outside (red border) the area. The computer simulation used integers instead of colors for the layout of the task space. The colors in Figure 1 are only used for the reader’s benefit in order to understand the experiment workspace. Agents evolved by ANT occupy 4 squares in the 2D simulation environment. The fitness of an agent is the number of resources collected in the
designated area in a finite number of time steps, $T = 300$. This evaluation is done using 50 such workspaces that differ in the distribution of resources. The inputs to an agent are as shown in Table 1.

<table>
<thead>
<tr>
<th>Sensory Input</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_1 \ldots V_4$</td>
<td>Resource Detection</td>
<td>Resource, No Resource</td>
</tr>
<tr>
<td>$C_1 \ldots C_4$</td>
<td>Workspace detection</td>
<td>Blue, Red, Orange, Floor</td>
</tr>
<tr>
<td>$S_1, S_2$</td>
<td>Obstacle detection</td>
<td>Obstacle, No obstacle</td>
</tr>
<tr>
<td>$LP_1$</td>
<td>Light Position</td>
<td>Left, Right, Center</td>
</tr>
<tr>
<td>$LD_1$</td>
<td>Light Range</td>
<td>0-10 (distance from the light)</td>
</tr>
</tbody>
</table>

Table 1: Sensory inputs for the task [Thangavelautham et al. 2007].

Figure 12 diagrammatically describes the sensory inputs for resource detection ($V_i \in \{0,1\}$), workspace detection ($C_j \in \{0,1,2,3\}$), and obstacle detection ($S_k \in \{0,1\}$).

Figure 12: Sensory inputs: a) resource detection, b) workspace color template detection, and obstacle detection [Thangavelautham et al., 2007].

The behaviors performed to execute the task are illustrated in Table 2. Within a single time step, these behaviors are activated based on the ANT controller’s output.

<table>
<thead>
<tr>
<th>Behavior order</th>
<th>Behavior</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Dump Resource</td>
<td>Move one grid square back; turn left</td>
</tr>
</tbody>
</table>
2. Move Forward Move one grid square forward
3. Turn Right Turn 90° right
4. Turn Left Turn 90° left
5, 7, 9, 11. Bit Set Set memory bit i to 1, i=1…4, respectively
6, 8, 10, 12. Bit Clear Set memory bit i to 0, i=1…4, respectively

Table 2: Behaviors to execute the task [Thangavelautham et al., 2007].

2.2 Evolution

The evolutionary algorithm is Darwinian in nature, similar to genetic algorithm described in section 1.3.1, that initializes a population \( P = 100 \) tissues, with a crossover probability of 0.7, mutation rate of 0.01, and a tournament selection of 0.06\( P \). Each tissue is initialized with 3 x 6 motor neurons in one layer grown to include 70-140 neurons, whose genetic information, as introduced in section 1.3.3, is obtained from a uniform random distribution. The structure and evolution of ANT controller has been coded in C language obtained from Dr. Jekanthan Thangavelautham, University of Toronto Institute for Aerospace Studies.

2.3 Learning

As mentioned in the introduction, five cases have been considered to comprehend the effects of learning on evolution. Table 3 summarizes the list of values for learning parameters used for the different cases.

<table>
<thead>
<tr>
<th>Case</th>
<th>Parameter</th>
<th>Type of learning</th>
<th>Environment</th>
<th>Training period</th>
<th>Discount (( \gamma ))</th>
<th>( \varepsilon )-greedy policy (( \varepsilon ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td></td>
<td>Q-Learning, SARSA</td>
<td>Moderate Figure 1</td>
<td>1/4th lifetime</td>
<td>1.0</td>
<td>5.0%</td>
</tr>
</tbody>
</table>
Table 3: Summary of parameters used for the five cases.

**Case 1:** Effect of type of learning method on evolution.

In this case, the learning algorithms incorporated in the lifetime of an agent being controlled by ANT are the SARSA TD reinforcement learning as explained in section 1.3.2.1 and Q-Learning TD reinforcement learning as explained in section 1.3.2.2. However, $Q(s, a)$ is not initialized arbitrarily. A copy of the evolved weights for the decision and motor neurons is made, hypothesizing one to one mapping from genotype to phenotype. This copy of weights and the initialized sensory inputs, $s$ (refer to Table 1), are fed through the feedforward network and coarse coding as explained in section 1.3.3.1. The behaviors ($a_i$, where $i=1\ldots12$ corresponding to behaviors mentioned in Table 2) activated by the feedforward network and coarse coding are used to initialize the corresponding $Q(s, a)$ to 0.5 and the rest are initialized to 0. The matrix $Q(s, a)$ is grown dynamically for each set of sensory inputs at every time step. An action is performed using these acquired $Q(s, a)$ values and policy. For this experiment, the $\varepsilon$-greedy policy is used as it is the most commonly used policy and is also easier to use when a
combination of behaviors is required. $\varepsilon = 0.05$, that implies random action is taken at the rate of 5%. As mentioned in the SARSA learning algorithm in section 1.3.2.1, $Q(s, a)$ is updated using

$$Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma Q(s', a') - Q(s, a)]$$  \hspace{1cm} (1.5)

Let $\Delta Q = r + \gamma Q(s', a') - Q(s, a)$  \hspace{1cm} (1.6)

Similarly, in $Q$-Learning algorithm in section 1.3.2.2, $Q(s, a)$ is updated using

$$Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$  \hspace{1cm} (1.7)

Where, $\Delta Q = r + \gamma \max_{a'} Q(s', a') - Q(s, a)$  \hspace{1cm} (1.8)

Reward $r=1$ if there is an increase in the number of resources in the designated area (except the resources held by the agents in case they are inside the designated area at a given time step) as compared to the previous time step, else $r=0$. The discount term $\gamma=1$, implying 100% consideration of the estimated consequences of the action taken. $\alpha$ is the learning rate, which is evolved from one generation to another. $\alpha$ for each tissue is initialized in the first generation with values between $[0, 0.01]$ and is evolved with a crossover probability of 0.7 and mutation rate of 0.01.

$Q(s, a)$, thus estimated, is forced to remain within $[0,1]$.

$\Delta Q$ is used to update the phenotypic weights of the decision and motor neurons. A weight update rule similar to tri-hebbian rule [Rivest et al., 2004] is used to update the motor neuron weights. The change in weight $\Delta W$ is given as,

$$\Delta W = \alpha \psi_\lambda(\sigma_\lambda) \Delta Q$$  \hspace{1cm} (1.9)

Where, $\psi_\lambda(\sigma_\lambda)$ is the activation value of motor neuron calculated using threshold functions (1.1), and $\alpha$ is the learning rate.

The decision neuron weights are updated using a rule similar to e-gradient method [Rivest et al., 2004].

$$\Delta W = \alpha \Delta Q [\psi_{\mu,t-1}(\sigma_\mu) - \psi_{\mu,t}(\sigma_\mu)] + \alpha \psi_{\mu,t}(\sigma_\mu)(\bar{q}_t - q_t)$$  \hspace{1cm} (1.10)
Where, \( \psi_{\mu,t-1}(\sigma_\mu) \) is the activation value of decision neuron for sensory inputs at time step \( t-1 \), \( \psi_{\mu,t}(\sigma_\mu) \) is the activation value of decision neuron for sensory inputs at time step \( t \), \( q_t \) is the average activation value of all decision neurons at time step \( t \) that acts as an inhibitory neuron activity to minimize the covariance of the decision neuron weights such that they don’t move in the same direction uniformly causing an individual to get stuck at local optimum, \( \bar{q}_t \) is the exponential moving average of \( q_t \). The exponential moving average of \( q_t \) is given as,

\[
\bar{q}_t = \omega q_t + (1 - \omega)\bar{q}_{t-1}
\]

(1.11)

Where, \( \omega \in (0,1) \) is the smoothing factor. The smoothing factor value that is closer to 0 has a more smoothing effect, which gives less importance to recent averages. There is no specific rule to choose \( \omega \). It is generally calculated as \( \omega = \frac{2}{T + 1} \). For \( T = 300 \), \( \omega = 0.007 \) considering \( \bar{q}_0 = q_0 \).

The decision and motor neuron weights are forced to remain within \([0,4]\).

During the lifetime of an agent, ANT controller is trained (weights are updated) for \( \frac{1}{4} T \) time steps and is tested (no weight updates) for the remaining \( \frac{3}{4} T \) time steps.

**Case 2:** Effect of change in environment on both pure evolution and evolution with learning

The change in environment is modeled by altering the location of the designated area for resource gathering. Two possibilities have been considered to study this effect.

**Case 2a:** Random change in environment.

The location, size and shape of the designated area change at random, yet confined by the blue border, for each generation. This represents a fast changing environment.

**Case 2b:** One-time change in environment

The location, size and shape of the designated area change as shown in Figure 13 after 250 generations.
Case 3: Effect of training period on evolution

A training period of $1/8^{th}$, $1/4^{th}$ and $3/4^{th}$ of lifetime is used to assess this effect. Refer to Table 3.

Case 4: Effect of discount rate on evolution

Three values have been considered for the discount rate: 0.3, 0.6, and 1.0. Refer to Table 3.

Case 5: Effect of learning policy on evolution

The effect of this learning parameter on evolution is studied by first carrying out the analysis with no allowance for any random action and second by allowing random actions at the rate of 5% during the agent’s lifetime. Refer to Table 3.
Chapter 3

3 Results and Discussion

This chapter states the results of the simulations as described in Chapter 2. We could have been optimistic that learning would accelerate evolution as defended by Baldwin [1896], and Hinton and Nowlan [1987]. However, Figure 15 to Figure 27 portray that although we have used biologically inspired learning methods, learning delays the performance of an individual being evolved using ANT.

Hinton and Nowlan [1987] argued that learning can accelerate evolution. Hence, although at the end of 500 generations learning individuals do not fare as well as those evolved without learning, learning should have accelerated the fitness values in the earlier generations. However, Figure 15 to Figure 27 show that learning instead decelerates evolution by producing individuals with lower fitness values. Also, as per Hinton and Nowlan’s study in 1987, learning can benefit individuals with lower innate fitness values by giving them an opportunity to improve their fitness values during their lifetime, hence smoothening the fitness curve and making it easier to find the optimum combination of alleles that will produce the maximum fitness (See Figure 2). Thus, we would like to expect a better average fitness values from the learning individuals. However, results plotted in Figures 16, 18, 20, 22, 25 and 27 discourage us from expecting a better average fitness.

Also, although Nolfi and Floreano’s [2000] claim that selection prefers individuals from the correlated environment is validated by Figure 14 (the existence of individuals with learning rates greater than 0, i.e. individuals perform well on the evolutionary and learning fitness landscapes), evolution with learning only manages to reach a local optimum.
These results imply that the costs of learning are overshadowing the benefits of learning. Mayley [1996] had listed these costs of learning. The simulation results incline towards the possibility of incurrence of a time cost. The learning method is based on a step wise reward accumulation (increase in fitness). As selection is governed by searching for the individuals with the best fitness at the end of an episode of 300 time steps and 50 different workspace boards, focusing on short term immediate rewards may not eventually contribute to the final fitness value for that individual. This results in a time cost which is quantified in Figure 15, where individuals following pure evolution reach a fitness value earlier (e.g. fitness = 20 in less than 50 generations) than those individuals evolved with learning (e.g. fitness = 20 in ~100 generations). For example, an agent moves forward and enters the designated area to dump the resources, because it is still holding the resources to be dumped at that given time step, the fitness calculator does not consider the resources held by the agent, thus there is no immediate reward obtained for this action. This may discourage the agent from performing this action which is necessary to contribute to the final fitness value of the individual.
Also, Mayley’s claim that the learning rate will eventually disappear from evolution as the generations progress proves to be valid for the evolution with learning cases. See the blue plot in Figure 14.

Learning rate represents plasticity of an individual that gives an individual the opportunity to change its neural network connections between the sensory inputs and output behaviors. As we can notice, although the learning rates decrease as the generations progress, they don’t disappear completely. In fact, during early generations natural selection prefers individuals with higher learning rates. This implies that learning is playing a role in improving the fitness of the evolving individuals. However, the cost of learning does not allow the individuals to perform as good as their innate capabilities.

However, as evolution moves towards finding the optimum individual it does not require much learning. Thus, the plasticity of individuals decreases with an increase in generations.

The following subsections discuss in detail the effects of change in environment and learning parameters on evolution.

3.1 Case 1: Effect of type of learning on evolution

In this case, the two types of TD learning method were used: SARSA (Section 1.3.2.1) and Q-learning (Section 1.3.2.2). As understood from the cliff walking example by Sutton and Barto [1998], SARSA takes a safer but longer path by following the state-action selection policy. On the other hand, Q-learning learns for an optimal path closer to cliff thus a shorter unsafe path, as it keeps falling off the cliff. Sutton and Barto conclude that SARSA learning performs better than Q-learning method for this task.
Figure 15: Maximum fitness for evolution vs evolution + SARSA vs evolution + Q-learning (Case 1)

Figure 16: Average fitness for evolution vs evolution + Q-learning vs evolution + SARSA (Case 1)
However, Figure 15 shows that Q-learning performs 9% better than SARSA. This implies that Q-learning reduces the time-wasting cost of learning.

Yet, as seen in Figure 15 the average maximum fitness of individuals obtained by pure evolution is about 30% better than the average maximum fitness of individuals that incorporated learning during their lifetime as they evolved over 500 generations. Thus, Q-learning does not minimize the time cost enough to improve the maximum fitness of the individuals to surpass the fitness of the individuals obtained by pure evolution.

3.2 Case 2: Effect of change in environment on evolution and evolution with learning

The environmental conditions play a very vital role in assessing the performance of the controller. The environment for the other cases changes moderately, i.e. each generation is tested on 50 different boards that have varying distribution of resources. This case studies the effect of changing the environment randomly for each generation and a one-time sudden change in the environment after 250 generations.

3.2.1 Case 2a: Effect of randomly changing environment on evolution and evolution with learning

A randomly changing environment that occurs by changing the location, size and shape of the resource gathering designated area, represents a fast changing environment. This is a rigorous testing criterion for both controllers, ANT evolved with learning and ANT evolved without learning, as the controller will need to predict and map itself with the random change in the environment over the generations.
Figure 17: Maximum fitness for evolution vs evolution + learning in randomly changing environment (Case 2a)

Figure 18: Average fitness for evolution vs evolution + learning in randomly changing environment (Case 2a)
Figure 17 and Figure 18 show that pure evolution again beats evolution with learning in an attempt to plan itself to handle the change in the environment for each generation. Though with larger noise, there is an upward trend in the maximum and average fitness values of individuals obtained by pure evolution. Fitness values of individuals evolved with learning during their lifetime improves very poorly as compared to pure evolution.

One reason for poor performance of learning individuals is that their inherited genes are weaker than the genes inherited by individuals obtained by pure evolution. The weaker genes need to optimize their weights during their lifetime which helps them slightly improve their fitness over the generations, however not as good as the purely evolving individuals. Figure 27 shows that if plasticity (learning) is participating in the evolution process, then natural selection prefers individuals with higher learning rates to help improve the performance of the weak genes inherited. This results in a deceleration of evolution.

3.2.2 Case 2b: Effect of one-time change in environment on evolution and evolution with learning

This case is not as tough as Case 2b. In this case, the environment changes once after 250 generations. After evolving individuals for 250 generations, in the case of pure evolution most genes present in the gene pool are specific to the task space they have been evolved for, thus reducing diversity. This results in a dip in the fitness curve, as the individuals will now need to adapt and evolve in the new environment. Though the fitness values are recovered soon (within ~25 generations), the increase in uniformity of the genes before the change in environment leads to fitness values lower than that obtained shortly before the change. See Figure 19 and Figure 20.
Figure 19: Maximum fitness for evolution vs evolution + learning in one-time changing environment (Case 2b)

Figure 20: Average fitness for evolution vs evolution + learning in a one-time changing environment (Case 2b)
Also, we can see in Figure 19 maximum fitness values for individuals learning as they evolve through the generations are not as competent as purely evolving individuals. However, after the dip, the learning individuals eventually catch up with purely evolved individuals. Moreover, they exceed in fitness value as compared to before the change in environment. This is because individuals obtained after the incorporation of learning offer variety of genes to nature to select from that can suit the individual’s ability to adapt the change in environment. This is verified by the average fitness values of individuals that learn to adapt to the change in environment eventually doing as well as the purely evolving individuals (See Figure 20). Figure 14 shows a decrease in learning rate after the change in environment that indicates natural selection prefers individuals with lower learning rates that are available in the population when the change in environment occurs resulting in fitness values approximating fitness values obtained from pure evolution.

3.3 Case 3: Effect of training period on evolution

![Maximum Fitness for Case 4 (average over 5 runs)](image)

Figure 21: Maximum fitness for evolution vs evolution + learning for different training periods (Case 3)
As we notice in Figure 21 and Figure 22, a training period of 1/4th lifetime results in higher fitness values as compared to fitness values obtained for a training period of 1/8th lifetime and 3/4th lifetime. For any kind of learning, supervised, reinforcement or unsupervised, in an attempt to find a curve approximating a desired curve, lesser training periods produce a poor of the curve. On the other hand very high training periods lead to over-fitting of the curve. See Figure 23 from Bishop’s Pattern recognition and machine learning [2006], page 7, as an example for poor and over fitting of a curve. Poor fit does not provide enough information to approximate the desired curve, whereas over fitting fine tunes the results to match with each data point thus causes it to consider the noise in the data points. This produces erroneous test results.
Similarly, training the ANT for 1/8th of an agent's lifetime was not enough to approximate a neural network better than the individuals obtained after training ANT for 1/4th of the agent's lifetime resulting in individuals trained for wrong behaviors. On the other hand, training ANT for 3/4th of the agent's lifetime resulted in overfitting. Over fitting resulted in surpassing the optimum weights for the network resulting in wrong behaviors and time-wasting costs.

### 3.4 Case 4: Effect of discount rate on evolution

Discount rate represents the percentage of resulting state-action value to be considered to evaluate the error in the current state-action value in order to update the latter. This error value (equation (1.8)) is in turn used to update the network neuron and decision neuron weights (equation (1.9) and equation (1.10), respectively).
Figure 24: Maximum fitness for evolution vs evolution + learning for different discount rates (Case 4)

Figure 25: Average fitness for evolution vs evolution + learning for different discount rates (Case 4)
It is noticed in Figure 24 and Figure 25 that evolution prefers a discount rate of 100% implying a complete consideration of resulting state in order to strengthen or weaken the neuron weights in an attempt to establish better networks between sensory inputs and output behavior.

The lower fitness values for lower discount rates can be justified by assessing equation (1.8). Lower discount rates can result in depreciation of the current state-action value even if the resulting state has the capability of delivering better results. Also, if a reward is earned, by not considering the total benefits of the resulting state, the weights of the current network will not be strengthened enough to optimize the network between the sensory inputs and output behaviors. This may eventually result in learning wrong behaviors and incur a time-wasting cost as lower discount learning methods seek the optimum neural network.

3.5 Case 5: Effect of learning policy on evolution

For the experiments in this thesis the $\epsilon$-greedy policy has been used. The $\epsilon$ value determines the percentage of lifetime an individual performs a random action. This value can prevent the learning method to stall at a local optimum.

![Figure 26: Maximum fitness for evolution vs evolution + learning for different $\epsilon$ value in $\epsilon$-greedy policy (Case 5)](image-url)

Figure 26: Maximum fitness for evolution vs evolution + learning for different $\epsilon$ value in $\epsilon$-greedy policy (Case 5)
Figure 27: Average fitness for evolution vs evolution + learning for different $\epsilon$ value in $\epsilon$-greedy policy (Case 5)

However, Figure 26 and Figure 27 show that the existence or omittance of this value does not pose any significant change in effect of learning on evolution. This implies that the stall by the incorporation of learning method is not due to the policy used.
4 Conclusion and future research work

The maximum fitness obtained for different cases described in Chapter 2 and discussed in Chapter 3 (except Case 2) have been summarized in Figure 28. As we see in the bar graph optimizing values for learning parameters by trial and error, and allowing the learning rate to evolve over generations does not help individuals learning during their lifetime evolve to match the individuals obtained from pure evolution. The best set of learning parameters obtained after testing all cases for 5 runs each are: learning rate = evolved (Figure 14), discount rate = 1.0, training period = 1/4\textsuperscript{th} of lifetime, $\epsilon = 5.0$.

![Figure 28: Maximum fitness values for Case 1, 3, 4 and 5 vs evolution](image)

Changing the environment in Case 2 shows that learning benefits evolving individuals only if there is a sudden change in environment due to the diversity present in the gene pool, however, the resulting fitness values match with those evaluated from the performance of purely evolved
individuals that witness a drop in their fitness values after the change due to reduced diversity in their gene pool. This helps us conclude that in an environment that changes moderately (using 50 boards with random distribution of resources), one time change in environment (one-time change in location of designated area for resource gathering), and randomly changing environment (random location of the designated area for every generation) although we use a biologically inspired controller and learning methods with best selected learning parameters, learning delays evolution due to a time cost. There is also a cost of learning wrong behaviors as the learning individuals fail to learn delayed gratification by focusing on earning immediate rewards rather than focusing on long term fitness evaluations.

The results of the five cases and their respective effects on evolution as discussed in Chapter 3 incline towards Mayley’s [1996] conclusion that the costs of learning overshadow the benefits of learning, thus decelerating evolution. Although, Baldwin [1896] and Waddington [1942] propose genetic assimilation of learned traits, these traits result in locally optimum individuals.

This conclusion makes us ponder that are we (Homo sapiens), who depend on learning with a training period of 20 – 30 years and its application for the rest of our life, optimized individuals, or is there hope for us to achieve optimum, or are we stuck at local optimum? The above results discourage us from answering the former two questions in the affirmative.

Although, this thesis study concludes that the costs of learning overshadow the benefits of learning, further research on improved biologically inspired learning methods and their effects on evolution can be assessed. Also, the performance of a controller can be examined by evolving it for one task and then placing it in a different task workspace to perform a similar task. A comparison can be drawn between the fitness values obtained for testing the controller with and without learning. The performance can be expected to be similar to having a cricketer play baseball.


