5-2013

Evolution of Complex Features in Digital Organisms

Javier A. Magana
University of Texas-Pan American

EVOLUTION OF COMPLEX FEATURES IN DIGITAL ORGANISMS

A Thesis

by

JAVIER A. MAGANA

Submitted to the Graduate School of the
University of Texas-Pan American
In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2013

Major Subject: Computer Science

EVOLUTION OF COMPLEX FEATURES IN DIGITAL ORGANISMS

A Thesis
by
JAVIER A. MAGANA

COMMITTEE MEMBERS

Dr. Laura Grabowski
Chair of Committee

Dr. Richard Fowler
Committee Member

Dr. Pearl Brazier
Committee Member

Dr. Emmett Tomai
Committee Member

May 2013

ABSTRACT

Magana, Javier A., Evolution of Complex Features in Digital Organisms. Master of Science (MS), May, 2013, 85 pp., 26 tables, 38 figures, references, 14 titles.

This thesis examines how complex features evolve in digital organisms, and how capabilities that evolved earlier provide a scaffold for evolving new features. I wanted to see how the organisms used previous adaptations to succeed in new environments, and how significant are these previous adaptations. The system used is the Avida platform, which is software that implements Darwinian evolution on self-replicating digital organisms. First, the system is seeded with ancestors that can only replicate, which gave a baseline for the quality of the task at the end of evolution. Second, the system used ancestors from a simpler environment as seeds. I observed that the quality of the task improved, but not drastically except for one of the environments. Lastly, the organism gets transferred form the simplest environment to the most complex one. I observed that only when the transplant included a similar complex adaptation the improvement was remarkable.

DEDICATION

The completion of my graduate studies would not have been possible without the support of my advisor, Dr. Laura Grabowski, whose guidance helped me when I was lost.

My friends, Maria Garza, Edgar Cantu, Francisco Alanis, Carlos Gomez, Guillermo Cabrera, and all the people that helped me get where I am today.

My father Javier and my mother Laura for their love, support, acceptance, and a lot of patience.

For my daughter Alexandrea who opened my life to a whole new set of experiences, and finally, Priscilla, my wife, for all her support in the career I have chosen, for our first child, all her love, and above everything else, for making my life complete. I love you with all my heart.

ACKNOWLEDGMENTS

I will always be grateful to Dr. Laura Grabowski, chair of my thesis committee, for all her mentoring and advice, from research ideas, to experiment design and data processing. She encouraged me to complete this process with patience and guidance. My thanks go to my thesis committee members: Dr. Pearl Brazier, Dr. Richard Fowler, and Dr. Emmett Tomai. Their advice, input, and comments on my thesis helped to ensure the quality of my intellectual work.

I would also like to thank the entire Computer Science faculty. Their hard work inspired me, and challenged me to always be better.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

CHAPTER I

INTRODUCTION AND BACKGROUND

Introduction

Motivation

In order to survive, organisms have evolved many complex adaptations. Even simple organisms have complex functions and structures. Bacteria communicate using secreted molecules to coordinate the behavior of the group. Complex hierarchical regulatory circuits have evolved to integrate and process the sensory information (Brelles-Marino, 2001). Brains are a good example of complex adaptations. They extract patterns from a noisy, non-stationary, and often unpredictable environment. Brains control and coordinate movement, form memories, and construct models of the world and its dynamics (Koch, 1999).The brain is generally considered the most complex organ in animals since it drives the behavior of the organism.

All of the examples mentioned above deal with one type of complexity called *functional complexity*, meaning the degree of complexity of the realized function. By contrast, the complexity that deals with the amount of information the gene stores about its environment is called *gene complexity*. Functional complexity analysis can give an insight on how different genes group together to perform a common function, while analyzing the gene complexity will help understand how difficult would it be to get the correct sequence of instructions to evolve a successful adaptation. Through the analysis of both complexities, we also might be able to observe the influence of each type of complexity.

Central Issues and Questions

The evolution of complex features is a key issue in evolutionary biology. Insight into this issue both helps explain the presence of biological complexity and, provides inspiration to computer scientists and engineers. Investigating the evolution of complexity can provide guidance in how we create and use new technologies, including drugs, the Internet, and self-replicating robots. Trying to find effective ways to improve on these technologies is difficult, but if we can identify how basic elements affect complexity, it might be possible to create new and more effective solutions to complex problems.

Evolution has repeatedly produced highly complex traits. To understand the details of how evolution produces something as complex as the mammalian eye (Darwin, 1859), we would need a perfect fossil record, ideally including the genetic information for the entire line of descent. This ideal situation does not exist in the natural world, so such analysis is hard to perform in living organisms. Digital evolution, an approach within the larger discipline of artificial life, offers a potential solution to many of the problems involved in studying evolution in living organisms. Working with virtual environments, artificial life studies systems related to the processes and evolution of life, with the goal of understanding the principles of life. There are several advantages in using artificial life methods such as digital evolution. Digital evolution systems allow us to track evolution as it occurs, and produces the complete record we need for analysis. It makes it possible to analyze the population without disturbing other aspects in the environment, something that is impossible to do in natural systems without disturbing the environment. Investigators are also able to control most aspects of the environment of evolution so that hypotheses can be tested with proper experimental control. This degree of control is difficult to achieve with natural systems.

Digital evolution has provided insights into how complex features arise and the limitations to evolving complex features. In Lenski *et al.* (2003), digital organisms evolve nine logic functions (from simplest to most complex: not (NOT), not and (NAND), and (AND), or not (ORN), or (OR), and not (ANDN), negative or (NOR), exclusive or (XOR), equals (EQU)). An organism can execute faster depending on the complexity of the logic function that it can perform, and as a result the organisms can have more descendant in future populations. Fifty experiments were executed, and of these 23 experiments evolved the most complex logic function, logical equals (EQU). The sample organism analyzed that evolved to perform the EQU function had 60 instructions. If the experimenters removed any of 35 of these 60 instructions, the organism was not capable of performing the EQU function. This result demonstrated that there are functional relations between simpler and more complex functions. The study also included an experimental set up where only the EQU function was rewarded and allowed the organism to execute faster. The EQU instruction did not evolve in this setup because the organisms did not evolve the simpler functions required to construct the more complex EQU. This situation also caused the organism genome to be smaller, and demonstrated the validity of the hypothesis presented by Darwin that complex functions depend on basic ones.

In another study, (Ofria, 1999), digital organisms evolved instructions for multithreading that would allow the organisms to execute different portions of it genome to execute in parallel. They used three different environmental configurations with different selective pressures to see how each affects the evolution of the organisms. They used the first environment as control, where only the replication rate is the only selective pressure. The second environment included selective pressures to perform twelve logic functions. The more complex the function, the faster an organism could execute. The third environment used 80 logic operations as a selective

pressure. The investigators discovered that once an organism evolves the capability to use multiple threads, different portions of the genome (here, a set of instructions) handle different operations. Multiple threads can use the same portion of the genome, but this approach carries a greater risk because a change in this portion can be detrimental to the organism. This risk causes reduced variation for these portions of genomes. The researchers were able to analyze the variability of portions of the genomes, and they observed that the greater the rate of adaptation, the less the organisms used multiple threads.

In this thesis, I will explore the evolution of complexity through evolution in virtual environments that present different problems to be solved, as presented in Grabowski (2009). Digital organisms evolve in environments that require them to sense and react to cues in the environment to do well. The digital organisms receive directional cues to navigate the environment, and then "decide" if they want to do anything with the information received. The more simple cues indicate a turn either right or left, while a more complex one requires that the previous turn cue be remembered. I focus on the following questions in my study.

How difficult is it to evolve the behavior from a simple ancestor organism in the different environments? Organisms evolve in different environments of increasing complexity. By starting evolution from the same initial ancestor in each environment, I will examine how the different environmental cues and configurations affect the evolution of the organisms.

Do features or capabilities that evolve earlier provide a scaffold for building a new feature? As demonstrated by Lenski *et al.* (2003), simpler functions may be used to evolve functions that are more complex. An ancestor that successfully evolved tactics to traverse the environment might provide a scaffold for building new capabilities, giving an edge to its descendants for more complex traits and environments.

What genomic differences do the organisms that evolved from scratch and the ones that evolved to be successful in another environment have? In the study by Lenski et al. (2003), the genome of the organism shrank when it tried to perform the most complex logic function (EQU) without having evolved more basic logic functions. If we use an organism that was successful in a previous environment, and this organism produces a successful population in the new environment, the genome from these organisms should show the new genome instructions, and increase in the genome length because this increase in length diminishes the probability of replacement from one of the useful.

Background

Computational Approaches to Evolution

The origins of evolutionary computing can be traced back to the late 1950's (Friedberg, 1958), but it was relatively unknown to the broader scientific community. Use of evolutionary computation has steadily increased over the years. Evolutionary models provide advantages for some tasks because they gain flexibility and adaptability to constantly changing problems. Examples of application domains include neural networks with changing network structure, and fuzzy logic augmented by evolutionary computation provide robust search capabilities in a complex space (Cordon, 2004). Evolutionary computation should be considered as a general adaptable concept for problem solving, especially well-suited for solving difficult optimization problems, rather than a collection of related and ready to use algorithms (Back, 1997).

Evolutionary programming was introduced by Holland (Holland, 1962). It was originally offered as an attempt to create Artificial Intelligence. The approach was to evolve finite state machines (FSM) to predict events based on former observations. The performance of an FSM

with respect to its environment might then be measured based on the machine's prediction capability, *i.e.* by comparing each output symbol with the next input symbol and measuring the worth of a prediction by some payoff function (Holland, 1962).

Avida Overview

The Avida Digital Evolution platform was created in 1993 by Charles Ofria, Christoph Adami, and Titus Brown. Digital Evolution is a form of evolutionary computation that places a self-replicating computer program (called a *digital organism* or *Avidian*) in a computational environment. Avida provides the basic ingredients necessary for evolution: replication, competition and variation (Dennett, 2002). Most Avida experiments are seeded with a simple self-replicating ancestor, *i.e.*an organism that has only the ability to copy itself into a space of memory that will become its offspring. When the digital organisms copy themselves (*i.e.* self-replicate), like a computer virus, there is a possibility for variation. The variation, a mutation in Avida, can be an introduction, deletion, or change of Avida instructions in the offspring's genome. When an organism has finished its replication, its offspring genome is divided from that of the parent, and the offspring is placed in the environment, replacing any other organism that occupied that location. Thus, the fundamental competition in Avida is created by the limited space in the environment.

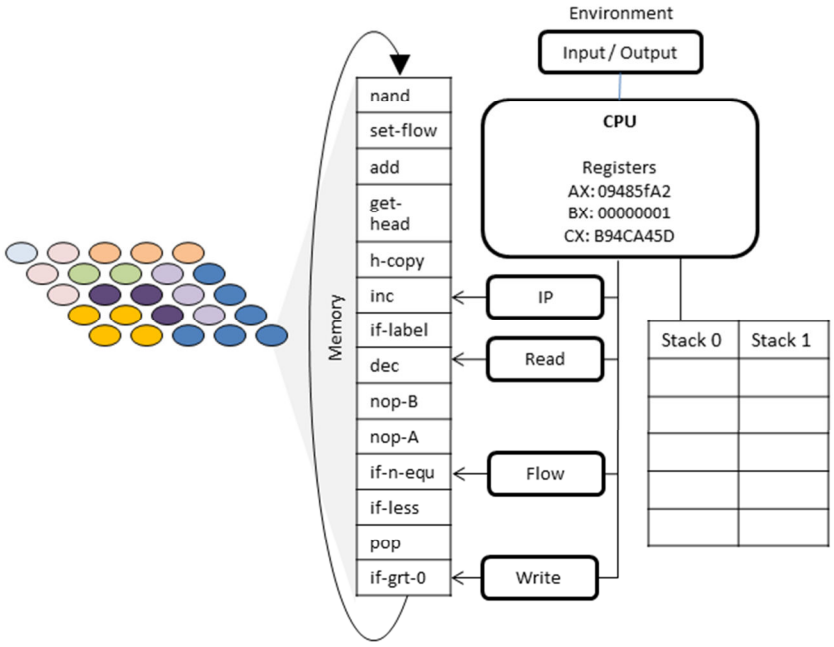


Figure 1. The standard virtual machine hardware in Avida: CPU, registers, stacks, heads, memory (genome), and environment I/O functionality.

The Avida world is a two-dimensional grid of cells. Each cell can only have one organism at a time. Each individual organism (Figure 1) is made of instructions, its "genome", and a virtual central processing unit (CPU). When the organism reaches the last instruction in its genome, execution goes back to the beginning in a circular fashion. The standard virtual CPU consists of three registers, two stacks, and four heads (FLOW, used as a jump target; IP, an instruction pointer that indicates what command is it going to execute next; READ and WRITE heads are used for organisms' replication). Each instruction in the genome modifies the virtual CPU, and the cost of executing the instructions is measured in virtual CPU cycles. The Avidians perform all internal operations and interact with the world through their instructions.

Mutations can occur when an Avidian replicates by copying its genome into a block of memory. This block will be the genome of the offspring. When there is an error in the process of the genome copy, it produces a genome different from the parent. There are different ways that a

mutation can happen. Instructions may be inserted, deleted, or changed according to user-defined probabilities. The newly created offspring is placed in a grid cell that is selected at random using one of several placement methods as configured by the user. If the cell was occupied, the new Avidian overwrites that organism that was in the cell. The space in the grid is limited; this causes competition between the organisms for this resource. An Avidian that can replicate faster has an advantage over slower replicators because the faster replicating organism will have a higher probability of more descendants in future population than a slower Avidian. Execution speed is variable, and is determined in part by an organism's "metabolic rate". The metabolic rate is used to allocate virtual CPU cycles, and a higher metabolic rate increases the speed the Avidian can execute instructions by allocating more virtual CPU cycles. For my experiments, I use an environment in Avida called the state grid. In the state grid each organism has separate information about its environment. The *state grid* is a virtual environment separate from the Avida standard environment. While a digital organism stays in the same place in the Avida world environment and does not move, each Avidian has a virtual grid where it can move independently of other organisms in the population. The only interaction between individuals in state grid experiments is when an offspring organism replaces another organism. The state grid allows simplification of the experimental design and implementation of experiments. Figure 3 shows a sample of a state grid. The state grid is easily defined and understood by human experimenters, with efficiency and relatively low computational overhead (Grabowski, 2010).

The state grids are defined in the Avida environment configuration file. The grids are torus shaped grid (doughnut shaped) that has different states defined by the user. The dimensions of the state grids are user defined, and they have no size constraints. The state grid definition also includes the organism's initial facing, location, and the state of each grid cell. Each cell has a

particular state that indicates the type of cue the organism will receive when it executes a sense instruction. The task the organisms have to perform involves traversing the environment. This task determines the metabolic rate bonus that an individual organism will receive when it replicates.

Each Avida organism has a facing (direction in which it is oriented). The organism must face a cell that is connected to the organism's cell. Because of differences between the geometries available in Avida, there are varying numbers of valid facings in certain geometries. In my experiments, I use a torus geometry, which creates an infinite plane with no edges because the top of the grid is connected to the bottom, and the right edge is connected to the left edge. In a torus geometry environment, all grid cells have eight valid facing directions (Figure 2).

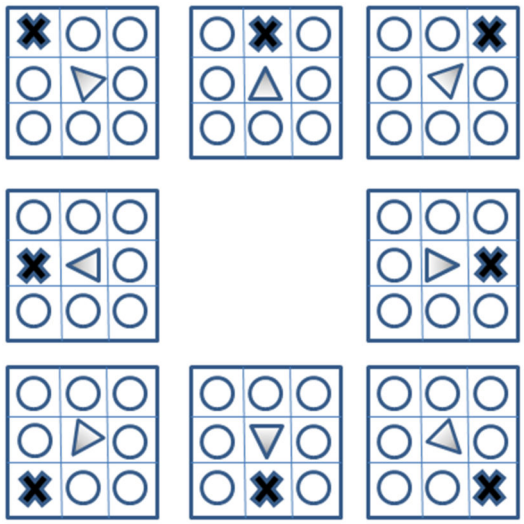


Figure 2. Valid orientations for an Avida organism in a torus grid (Grabowski, 2009).

CHAPTER II.

METHODS

Experimental Setup

The experiments are inspired by the maze-learning experiments with bees by Zhang *et al.* (1996). In these experiments, bees learned to fly through a complex maze by following a color marked trail. Bees that were trained to follow color marks through an initial part of the maze are immediately able to use the same sign-tracking cue to find their way through the rest of the maze, which was unfamiliar to them. Bees that were trained to follow color marks through a particular maze can use the same cues to negotiate an unfamiliar maze. Bees can learn to use color as a signal even when it indicates the path through the maze in a symbolic way, for example, blue indicating a turn to the right and a green a turn to the left (Zhang, 1996).

Avidians were used to study the origin of memory use in navigation in Grabowski (2009). She designed her environments to allow Avidians to traverse paths using environment sensory cues. The experiment configuration in Avida emulated the environments used for bees (Zhang, 1996). I will use the same environmental configuration to research how increasingly complex environments guide the evolution of the organisms.

State Grid Paths

Each state grid contains a single path for the Avidians to follow. The concept behind the environments is that an organism moves around the state grid environment, encountering different states that either increase or decrease the organism's "energy" by increasing or decreasing the metabolic bonus. When an organism moves along the path, it acquires more energy than the amount lost by performing the movement by encountering "food" states on the path. If an organism moves to a location off the path, it will not receive energy since those cells do not contain food. The longer the organism moves off the path, the greater the amount of energy wasted, reducing the overall metabolic rate bonus. Organisms that move along the food path build up energy, and are able to execute at an accelerated rate.

The environmental cues or signals that are present inside of the state grid paths vary depending on the experimental design. The following is a list of all the signals that are used:

Empty. This is a signal used to indicate that the organism is off the path. It is the only signal that does not provide a metabolic rate bonus.

Nutrient. It is the main component of the path. It indicates to the organism that is on the path. It contributes to the metabolic rate bonus only the first time that the organism occupies the grid cell.

Directional Cue. The directional cue is used to signal a turn where the path continues. The cue can be either left or right, and only by 45 degree increments. It is treated as a nutrient, and it contributes to the metabolic rate bonus.

General turn cue. This directional cue signals the repetition of the last encountered directional cue. It also contributes to the metabolic bonus.

Each experiment design uses different cues for the path. The following is a list of all the experimental sets used for each experiment, and the appropriate cues that are implemented per experimental set. For each type of environment, I used eight different path configurations to add more variation to the environments.

Single Turn. The first set of experiments uses state grids with paths that have turn cues in only one direction, right or left.

Right and Left. The second set used both left and right turn cues in the same environment.

Cue First. This is the third experimental set, and uses the first turn signal in the environment to specify the direction of the rest of the general turn cues (*i.e.,* right or left). This configuration is similar to the environment from the first set.

General Turn Cue. This is the fourth and last experimental set. It uses a combination of left and right turn signals, along with the general cue signals to guide the organism. In these environments, the explicit directional cue appears when the direction of turn changes from the preceding turn (*e.g.*, when the last turn was to the right and the current turn is to the left). When the turn is in the same direction as the preceding turn, the general turn cue appears.

Path traversal task

The path traversal task was defined in Grabowski (2009). The organisms receive a metabolic rate bonus for the path traversal task. The farther an organism advances on the path, the more bonuses it receives. If an organism travels the entire path without stepping off, it receives the maximum possible bonus. The task calculates the number of unique valid paths steps, and subtracts the steps into empty states:

$$Traversed = valid - empty \qquad \text{Equation 1}$$

where *valid* is the count of unique path cells encountered, and *empty* is the total count of empty states encountered. Each cell on the path is analogous to a "bread crumb"; once it is consumed, the organism does not receive any energy when it returns to the cell. This ensures that the organism stays on track, and avoids situations where organisms evolve the capability to just move back and forth between cells. Since organisms need to follow environmental cues, any movement off the path is considered a waste of energy.

The task quality (*TQ*) is calculated from how much of the path is traversed:

$$TQ = \left(\frac{traversed}{pathLength}\right) * processValue \qquad \text{Equation 2}$$

In Equation 2, *pathLength* is the total count of path cells (nutrients and directional cues) and *processValue* sets the bonus maximum and is chosen by the user.

Once the *TQ* value is obtained, it is used to determine the organism's metabolic rate bonus. The computation of the metabolic rate (*MR*) is:

$$MR = \begin{cases} (BaseRate) * 2^{0} \; if \; traversed \; \leq 0 \\ (BaseRate) * 2^{TQ} \; if \; traversed \; > 0 \end{cases} \qquad Equation \; 3$$

The *MR* is defined in a way so that it is always better for an organism to move ahead than to stay still. The organism receives the bonus exponentially to the task quality. The more the organism advances on the path, the more *MR* bonus it receives.

State grid instructions

There are four state grid instructions that an organism can evolve into its genome through mutations that allow organisms to sense and move in the state grid. The *sg-move* instruction moves the organism one grid cell in the direction it is facing. With the *sg-rotate-r* and *sg-rotate-l* instructions, the Avidian changes its facing by 45 degrees to the right or left, respectively.

Table 1. State values for the state grid experiments. The definition of a state grid includes a listing of the state of each cell in the grid. The *sg-sense* instruction returns the value shown for each state.

State	Return Value
Empty	-1
Nutrient	0
General Turn	1
Right Turn	2
Left Turn	4

The *sg-sense* instruction returns a value specified by the user, and it signals the cell state in the organism's current location. These state values are provided in the state grid definition. Table 1 lists the state values that are used for these experiments. The values that these instructions return are not used in any of the merit calculations. The values shown were used since they can be easily manipulated with the assembly-like code that Avida uses for the instruction set. The values are the same for all the experiments. This approach makes the organisms' tasks consistent throughout the experiments.

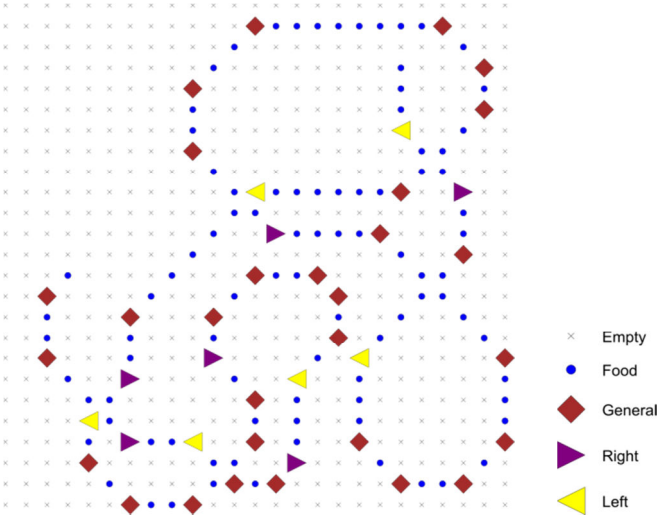


Figure 3. Graphical representation of a "State Grid" path with all the possible environmental cues.

There are two instructions that are used to compare values, the "if greater than" (*if-grt-X*) and the "if equal to" (*if-equ-X*) instructions. These instructions compare the value that is in the

BX register with a predefined value. The value to which is compared depends on the no-operation label (*nop*) that follows the instruction. Table 2 shows the comparison values.

Table 2. NOP-modified behavior of if-grt-X and if-equ-X instructions

NOP Label	Value for Comparison
Default (no NOP)	1
nop-A	-1
nop-B	2
nop-C	4

Evolution from a Simple Ancestor

The experiments use the state grid environment for the evolution of the digital organisms. This first set of experiments will be used to get a baseline of how the organisms evolve from scratch *(de novo)*, from an ancestor that only has the ability to replicate. As mentioned in Darwin's *On the origin of species* (Darwin, 1859), highly complex and specific organs cannot appear suddenly. They must evolve in incremental transitions through many intermediate states, and sometimes undergo changes in function. As mentioned previously, there are four sets of experiments in the current study: a single turn, right and left turns, cue first turns, and the general turn cue. They range from the simpler environmental configuration to the most complex one respectively. Each one will be use a simple organism that it is only able to replicate. Complex features are able to appear *de novo*, but organisms that evolve in stages, from simpler environments, should be more fit than organisms that have to deal with complex settings from the start.

Experimental Design

All the experiment sets have eight environments that the Avidian can be assigned to when it is born. For the single turn (ST) and cue first (CF) turn experiments there are four environments that have only right turns, and the other four have left turns only. For the ST experiments, the environment grids have three possible states: empty, nutrient, and either right or left directional cue (Figure 4). For the CF environment, the first directional cue that an organism encounters will be either left or right. The rest of the turn cues will use the general turn cue. The environment grids for the CF experiment have four possible states: empty, nutrient, general turn cue, and either right or left turn cue. This is similar to the single turn environments where all the turns are in one direction only for each path (Figure 5).

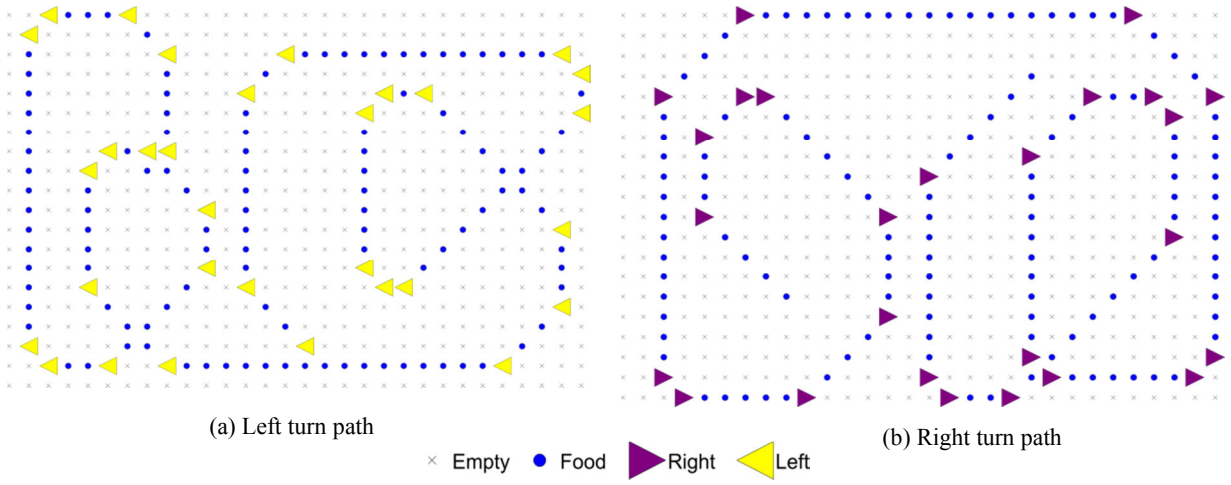


Figure 4. Sample "state grid" paths. These grids illustrate the single turn experiments. The left turn cue is represented by a yellow arrow. The Right turn cue is a purple arrow. The path is made of nutrients represented by a blue dot, and anything outside the path is empty represented by the gray X.

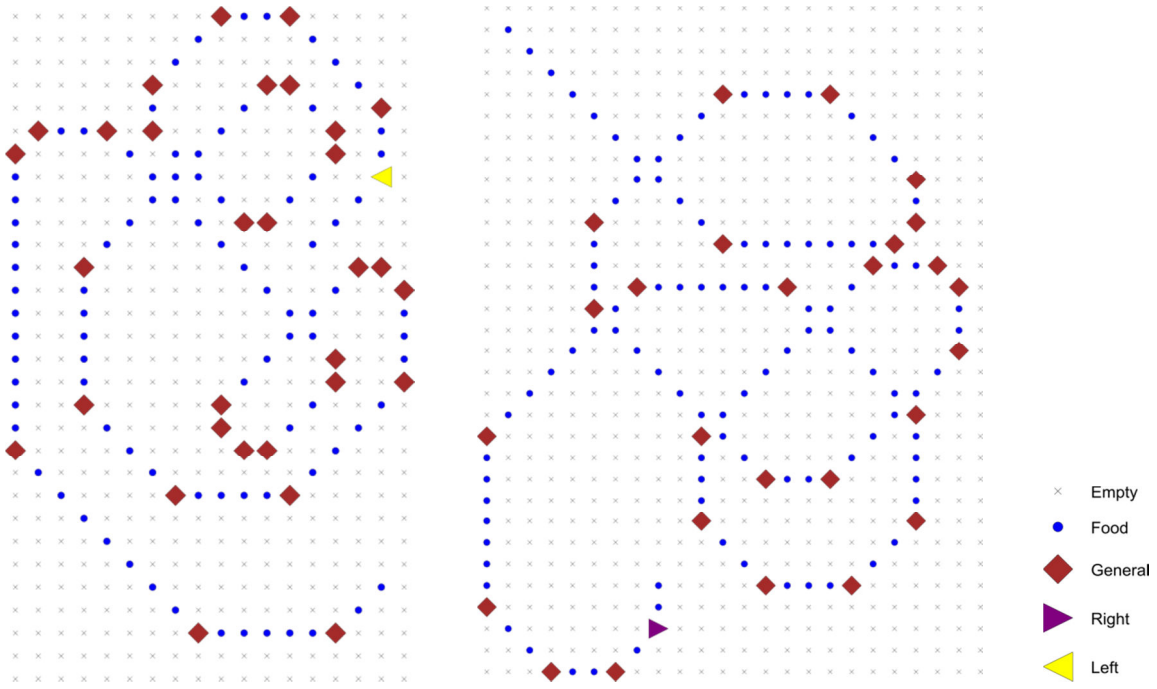


Figure 5. Sample "state grids" that illustrate the cue first environmental set up. The only absolute directional cue is the first turn encountered in the environment. The rest of the turns are made using the general turn cue. The first directional cue is used to indicate how the general turn cue will be interpreted. The purple arrow indicates a right turn. The yellow arrow indicates a left turn. The blue dot is a nutrient. The diamond is used as a general turn cue. The cells that are empty are represented by "x".

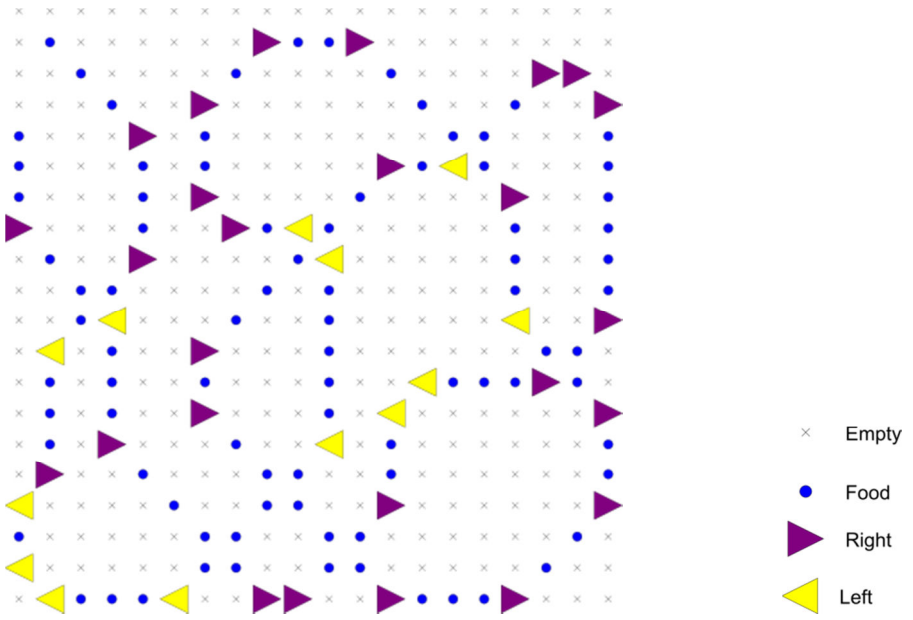


Figure 6. Sample "state grid" that illustrate the right and left environmental set up. The organism encounters both left and right turns through its life. The purple arrow indicates a right turn. The yellow arrow indicates a left turn. The blue dot is a nutrient, which is used to indicate the path. The cells that are empty are represented by "x".

The state grids for the right and left turns (RL) experiment have four possible states: empty, nutrient, right turn cue, and left turn cue (Figure 6). This environment is a more complex than the single turn, but it is simpler than the cue first environmental setup because the organism needs to be able to differentiate between the left and right cues during its lifetime. It contrast, the general turn cue environment (GT) is more complex than the RL environment because the organism needs to evolve a solution to remember the environmental cue encountered previously.

The last experiment set is the GT. For this experiment set the state grids' first turn is a directional cue, and the consecutive turns in the same direction following it use the general turn cue, until a turn in the opposite direction is found. This environment is the most complex one, and the state grid has all the possible states: empty, nutrient, left turn cue, right turn cue, and a general turn cue (Figure 7).

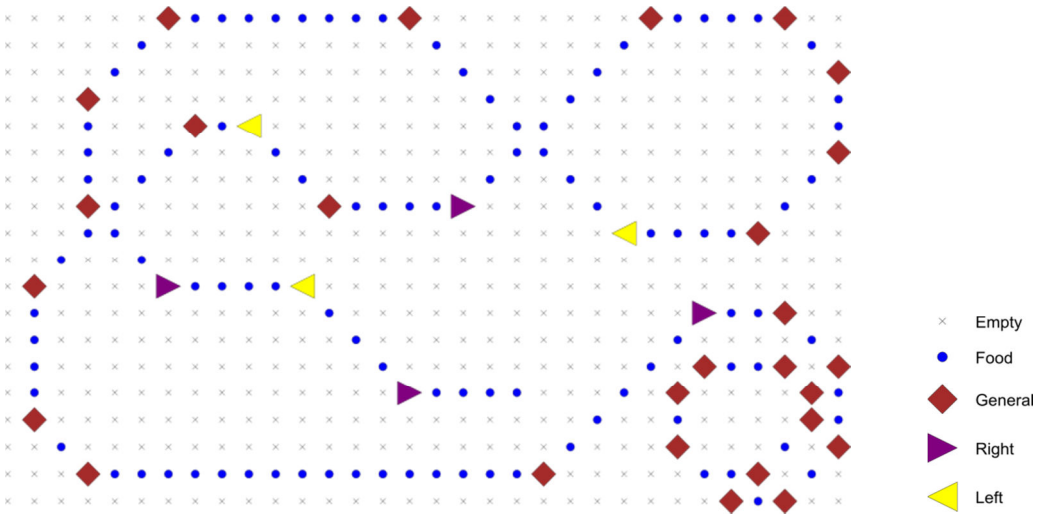


Figure 7. Sample "state grid" that illustrate the general turn cue environmental set up. The organism encounters both left and right turns combined with the general turn cue. The organism should evolve a way to "remember" the previous encountered directional turn cue, and use this information to turn in the right direction. The purple arrow indicates a right turn. The yellow arrow indicates a left turn. The brown diamond represents a general turn cue. The blue dot is a nutrient, which is used to indicate the path. The cells that are empty are represented by "x".

For each experimental set (ST, RL, CF, and GT) I ran 50 replicate experiments. Each experiment had a population maximum of 3,600 organisms (60 x 60 Avida world), and ran for 250,000 updates. The experiments were seeded with a default simple ancestor only capable of self-replication. The mutation rates used for the environment are the default Avida mutation rates, a 0.085 genomic mutation rate for an organism genome of length of 100 instructions. This includes a 0.0075 per site copy mutation rate, a 0.05 per divide insertion mutation rate, and 0.05 per divide deletion mutation rate (Ofria, 2002).

I used the task quality over time to evaluate the performance of the populations. The task quality measures the ratio of the total possible metabolic rate bonus an organism will receive based on its performance of the task, and is a direct measure of how much of the path an organism traverses without stepping off the path. I also used a behavioral test to see how the algorithm developed by the most fit populations performs when traversing a new path. This is done by running an execution trace of the evolved organism using unfamiliar grid configurations.

De Novo Experiments

Single Turn

Figure 8 shows the average maximum task quality (AMTQ) from all 50 replicate experiments for the single turn (ST) experiments. The organisms, on average, can traverse around half of the path by the end of the experiment. Figure 9 shows the distribution of the of AMTQ values for the ST experiment set per path. Although the medians for the values seem to vary for the different paths, there is no significant statistical difference between the AMTQ distribution for the paths at the end of evolution (Kruskal-Wallis Test, $p = 0.999$). Table 3 shows the top five ranking replicates from the *de novo* ST experiments with their task quality (TQ).

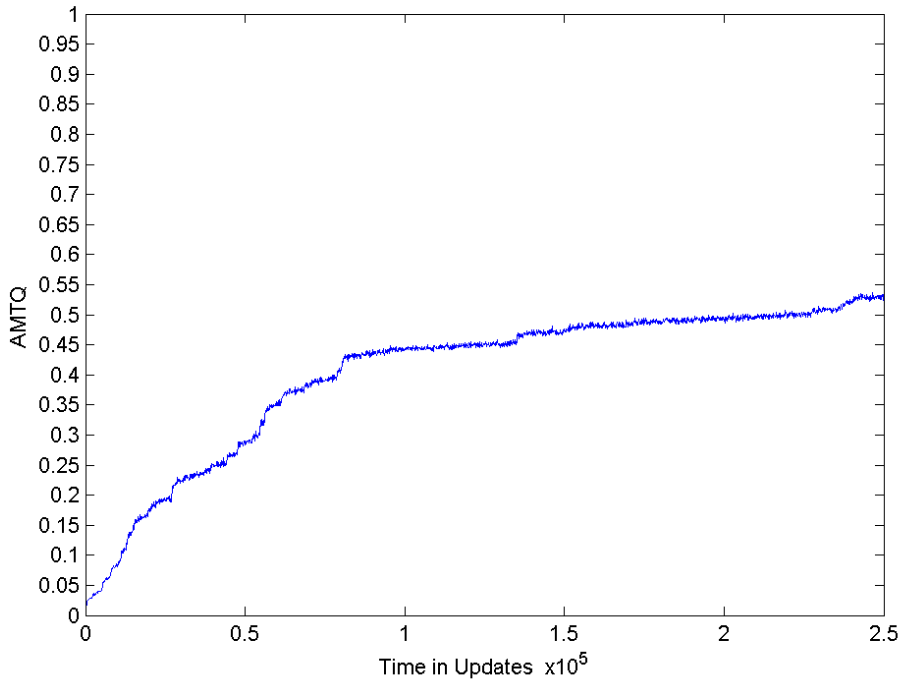


Figure 8. The average maximum task quality (AMTQ) for the ST experiment. The curve shows that on average the organisms can traverse about half of the paths by the end of evolution.

Table 3. Top five replicates from the de novo Single Turn experiments with their task quality (TQ).

Replicate	TQ
18	0.986024
7	0.985174
9	0.985174
22	0.985174
32	0.985174

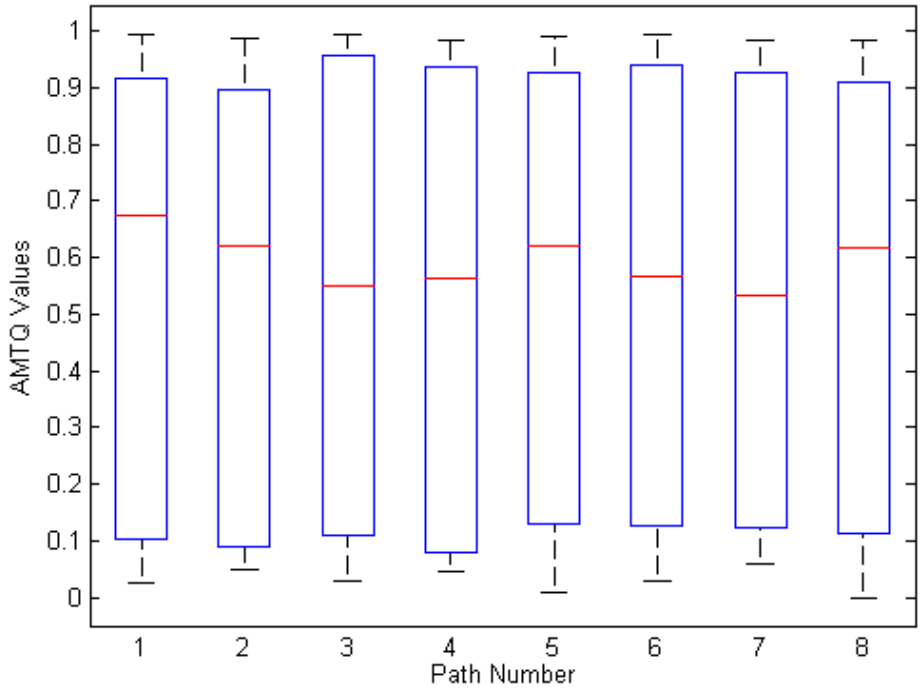


Figure 9. Distribution average maximum task quality per path. Odd path numbers are right turn only, and even paths are left turn only.

Figure 9 also shows that the most fit organism has an AMTQ very close to 1. The value is a direct indication of how much of the path an organism can traverse; the AMTQ means that the organism can traverse most of the environmental path. To test the behavior of this organism at the end of evolution, I ran a trace of the organism in two new paths (one with right turns only and one with left turns only) to make sure that the organism did not evolve a brute force solution to the paths encountered during evolution. This test will also help evaluate the strategy evolved by the organisms to traverse the path. An Avidian that evolved a generalized strategy for

traversing the paths should have no issues navigating a path that was not encountered during evolution.

The new path traversal results can be seen in Figures 10. These plots show that the top performing organism from the 50 replicate experiments was able to traverse both paths (right and left) successfully. In both cases, the organism stops moving after encountering an empty cell one step off the end.

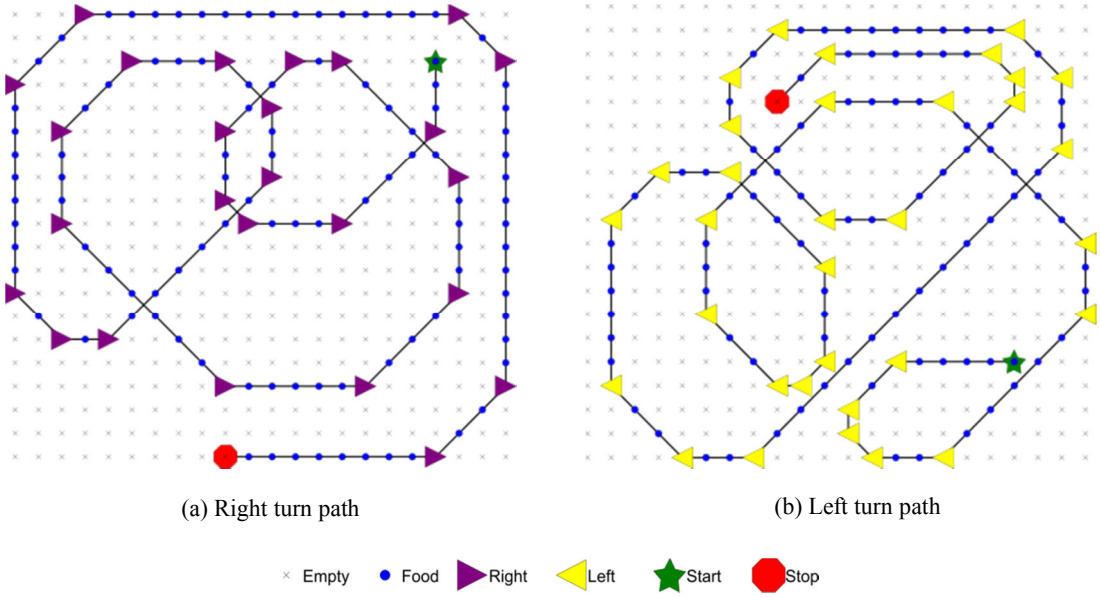


Figure 10. Trajectories of the most fit organism from the single turn experiment on paths that were not experienced during evolution. (a) The figure contains only right turns. (b) The figure contains only left turns. The green star represents the starting point. The organism final position is represented by a red octagon.

Figure 11 shows the result of a path that contains both left and right turns. The evolved organism's ancestors only encountered left turn cues or right turn cues in a lifetime, but the most successful organism was able to evolve a solution for a traversing a path that had both kind of turns in a single path.

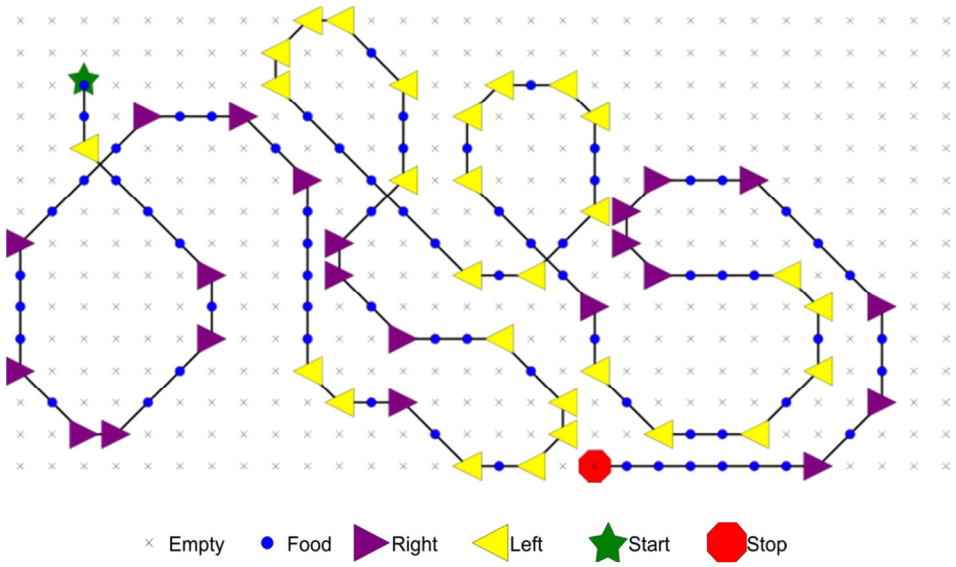


Figure 11. Trajectory of the most fit organism from the single turn experiment on a path that was not experienced during evolution. A path with both left and right turns cues are encountered through the organism lifetime. The green star represents the starting point. The organism final position is represented by a red octagon.

In order to understand how the organism is able to traverse the novel paths, I analyzed the execution traces for the organism as it executed. Most of the path following instructions and replication code are organized in a single module. The following pseudocode describes the functionality of the main module:

```
Do
      If (BX != -1)
            Rotate right
            Move
            BX = Sense
      Rotate left
      If (BX > 2) rotate left
      BX = BX / 2
      If (BX == 1) rotate right
      Copy
```

```
    Copy
While (no done copying genome instructions)
```

The first instruction in the module rotates the organism to the right. This instruction restores the original organism orientation when it is placed on the state grid. After this the organism moves straight ahead and senses the cue from the environment. It rotates left automatically after that in order to offset for the instruction at the beginning of the module. Then it checks if the cue is a left turn cue by comparing the value to be greater than 2. Table 2 shows that the left turn cue has a value of 4, which is the only possible value greater than 2. The organism makes two left turns when the cue is a left turn. To check if the sense value signals a right turn, the environmental cue, stored in the BX register, is divided by 2. When the result is 1, the organism makes a right turn. This restores the organism's original position, but once it goes back to the beginning of the module it rotates right again and moves. After that it copies two of the organism's instructions into the offspring. Once the organism senses an empty cue it stops moving, and continues replicating until finishes. Table 4 shows the Avida instructions for the main module described above. The main module consists of 22 instructions. The final genome size of the organism is 273 instructions. The genome size has almost tripled compared to the ancestor seed organism.

Table 4. Avida instructions for the main module of the top ranked organism in the ST experiment.

Main Module			
Position	Instruction	Position	Instruction
249	sg-rotate-r	260	shift-r
250	sg-move	261	if-equ-X
251	if-less	262	sg-rotate-r
252	sg-sense	263	if-label
253	sg-sense	264	h-copy
254	if-less	265	h-copy
255	h-search	266	if-label
256	sg-rotate-l	267	nop-C
257	if-grt-X	268	nop-A
258	nop-B	269	if-grt-X
259	sg-rotate-l	270	mov-head

Right and Left Turns

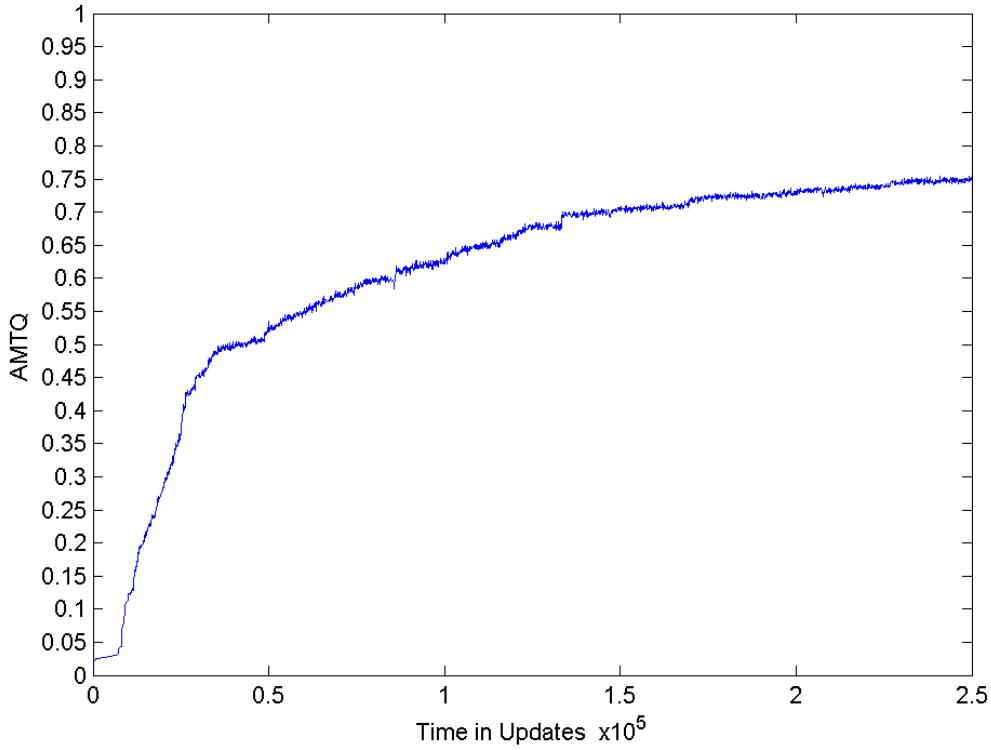


Figure 12. The average maximum task quality (AMTQ) for the RL experiment. The curve shows that on average the organisms can traverse about 75% of the path by the end of evolution.

Figure 12 shows the AMTQ from all 50 replicate experiments for the right and left (RL) environment. The organisms performed better than the ST organisms with an AMTQ close to 0.75 by the end of the experiment. This means that organisms are able to successfully traverse, on average, 75% of the path. This could be because of both turns are present in all environments instead of either one or the other. Figure 13 shows that the distribution of the AMTQ per organism has decreased overall, and that the median values have increased close to 0.95. There is no significant statistical difference in the performance on the different paths. (Kruskal-Wallis Test, $p = 0.999$). There are more low value performing organisms in the paths where the first directional cue is a right turn (Paths 3, 4, 6, and 8). The others (paths 1, 2, 5, and 7) have a left turn as the first directional cue. These last paths very seldom have low performing values. It is possible that the organisms from the experimental set evolved to perform left turns, but were not able to evolve right turns. Table 5 shows the top five performing replicates from the *de novo* RL experiment with their task quality (TQ).

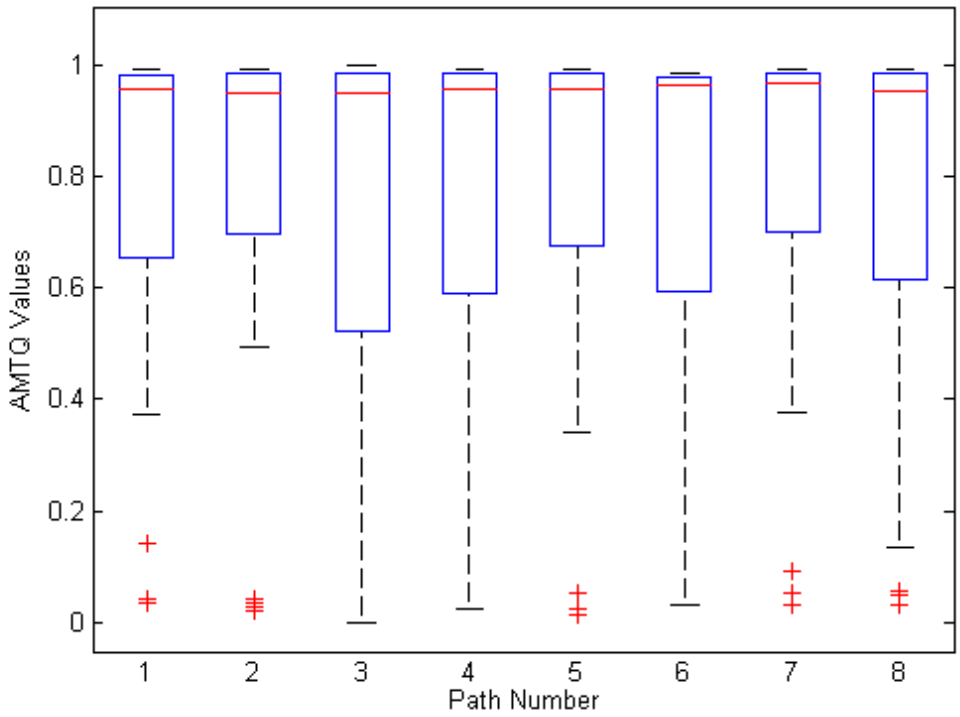


Figure 13. AMTQ value distribution for RL set. The RL experiment has higher median values compared to the ST set values.

Table 5. Top five replicates from the de novo Right and Left Turns experiment with their task quality (TQ).

Replicate	TQ
5	0.992289
12	0.985973
37	0.98474
3	0.983944
4	0.983944

Figure 14 shows the organism trace from the top ranked organism in the 50 RL experiments was able to successfully traverse paths with both directional cues to rotate to the left or right. The following is the pseudocode for the main module from this organism in the RL experiment.

```
Do

    If (BX != -1)
```

```
            BX = sense
            BX = BX/2
            If (BX > 1) rotate left
            If (BX == 1) rotate right
            move
      copy
While(finished copying genome)
```

This organism evolved a similar approach as the most fit organism in the single turn experiment. The organism senses the environment; if the cue is a left directional cue (value of 4) the value is divided in half, and checked to see if it is greater than 1. For the right directional cue (value of 2), the value also gets divided by 2, but now it is compared to 1. If both of these comparisons fail, the organism copies its next instruction to the offspring memory and moves. Once the organism encounters an empty cell it stops moving, and continues copying the genome for the child Avidian.

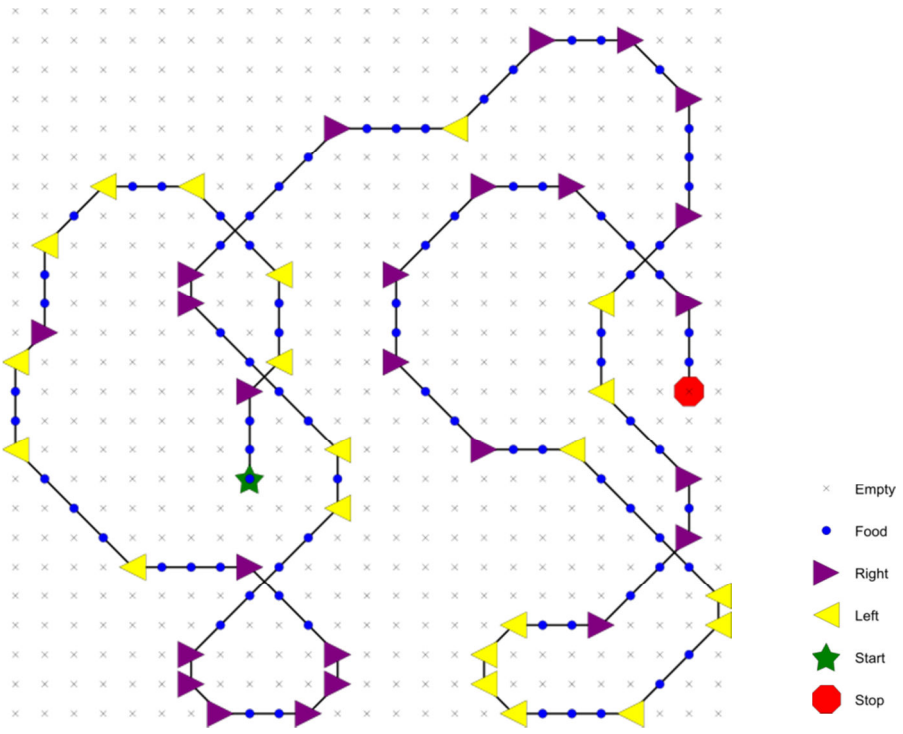
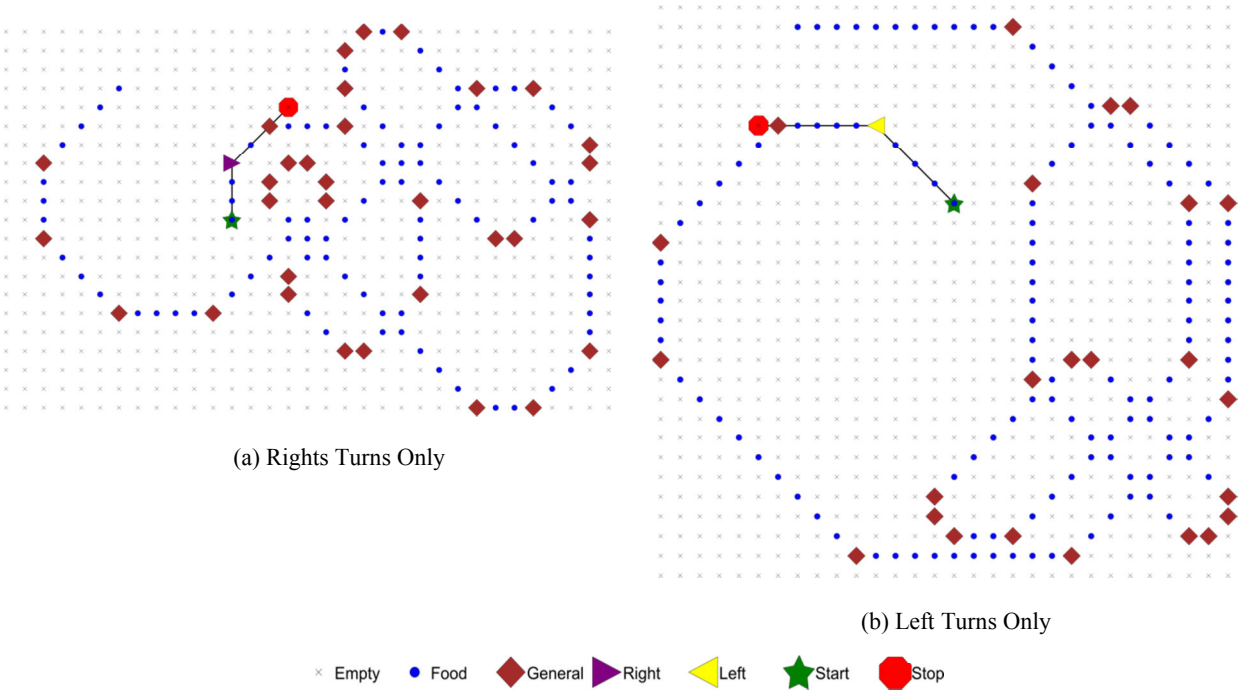


Figure 14. Trajectory of the most fit organism from the RL turns experiment on a novel path. The green star represents the starting point. The organism final position is represented by a red octagon.



(a) Rights Turns Only

(b) Left Turns Only

Figure 15. Trajectory of the most fit organism from the RL experiment on a CF environment. (a) Right turn only state grid. The first directional cue is a right turn. This should be used to represent the rest of the general turn cues. (b) Left turns only state grid. In this situation the general turn cue should be interpreted by the organism as a left turn. The green star represents the organisms starting point. The organism final position is represented by a red octagon.

The execution trace of the organism shows that its sensing capabilities are used to identify left and right turn cues. This means that the organism is not able to handle a general turn cue (Figure 15), which requires a more complex capability to evolve, and the general turn cue was not encountered during evolution. Table 6 shows the Avida instructions for the main module described above. The main module consists of 18 instructions. The final genome size of the dominant organism consists of 160 instructions.

Table 6. Avida instructions for the main module of the top ranked organism in the right and left experiment.

Main Module			
Position	Instruction	Position	Instruction
141	sg-sense	150	sg-rotate-r
142	shift-r	151	if-less
143	if-grt-X	152	sg-move
144	sg-rotate-l	153	if-label
145	get-head	154	nop-C
146	if-equ-X	155	nop-A
147	nop-A	156	h-divide
148	h-search	157	h-copy
149	if-equ-X	158	mov-head

Cue First Environment

On the cue first environment (CF) the Avidians need to evolve a solution to "remember" the first directional cue that they encounter in the path for the duration of their life. Figure 16 shows the AMTQ performance of the organisms is now reduced compared to the previous experiments. This might indicate that the organisms require more complex adaptations to deal with environmental paths. The plot also shows that, on average, an organism could only traverse around 30% of the path. Figure 17 shows the median values for the 50 replicate experiments is close to 0.2. This means that most organisms from this experimental set are only able to evolve to traverse 20% of the path. It also shows that the distribution of values has increased in range,

and although there are some organisms capable of traversing whole paths, this trait is becoming more unusual. In this environment there is no significant statistical difference in the performance by the organisms on the different paths (Kruskal-Wallis Test, p=0.933). Table 7 shows the top five ranking replicates from the *de novo* CF experiment set with their task quality (TQ).

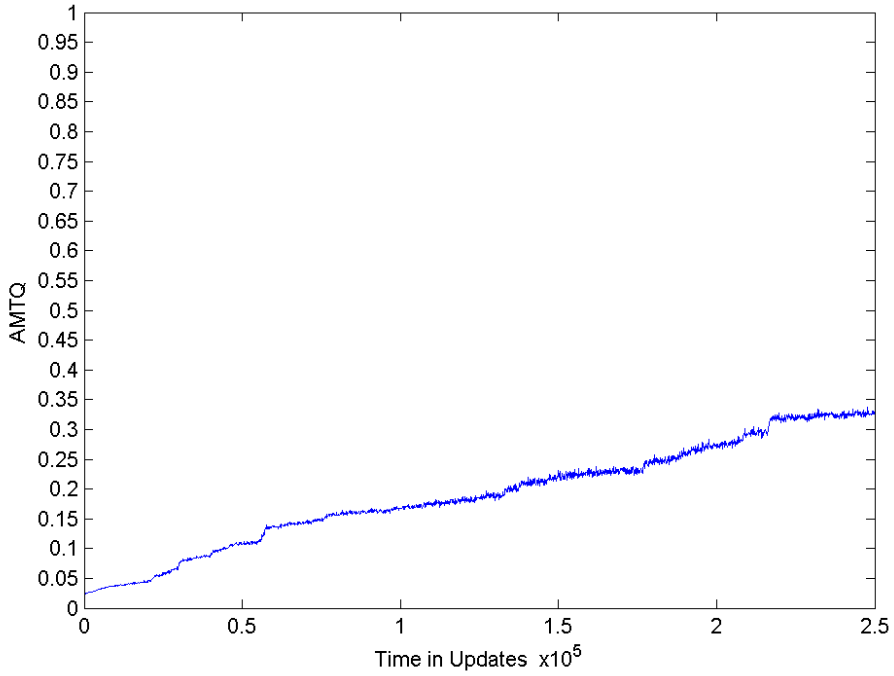


Figure 16. The average maximum task quality (AMTQ) for the CF experiment. The curve shows that, on average, the organisms can traverse about half of the paths by the end of evolution.

Table 7. Top five ranking replicates from the CF experiment with their task quality.

Replicate	TQ
5	0.986124
27	0.985285
8	0.87606
30	0.850264
22	0.803165

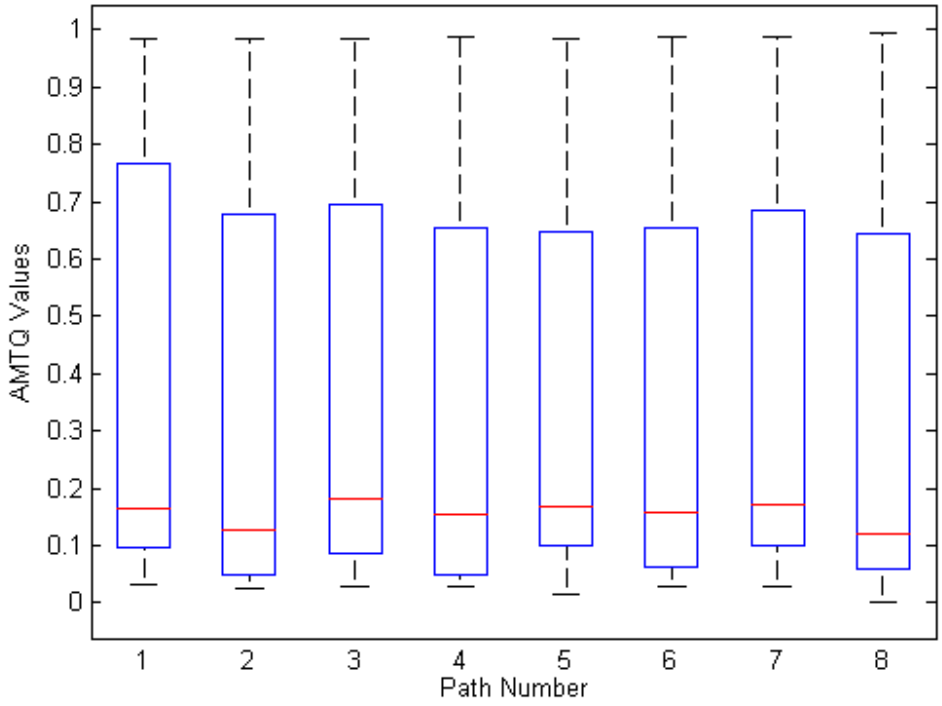
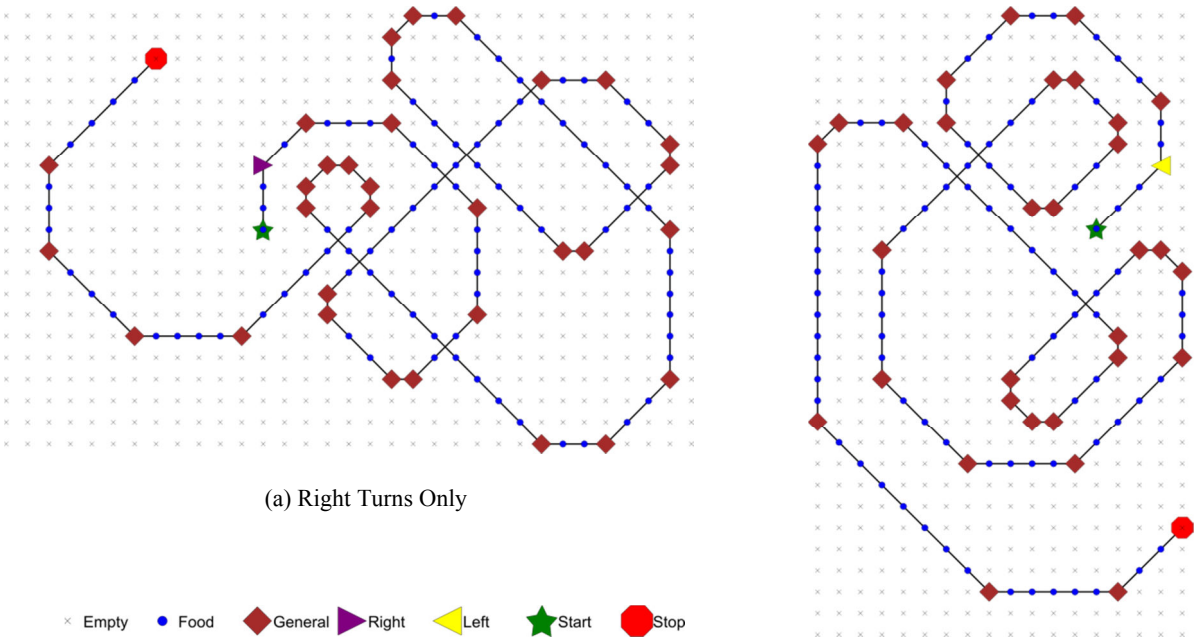


Figure 17. AMTQ value distribution for cue first experiment. This experimental set has a diminished median values compared to the right and left and single turn experimental set values. This means that the organisms now have difficulties evolving a solution to interpret the general turn cue.

Figure 18 shows the trace from the most fit organism. It can successfully traverse both environmental state grids where the first environmental cue is either a right or a left turn. The genome analysis shows that the organism evolved two separate modules to deal with each type of turn that was encountered in the environment. These modules are very similar in structure. They both have instructions for rotating, moving, and copying instructions; both modules are loops that will terminate the genome execution once replication is complete.



(a) Right Turns Only

(b) Left Turns Only

Figure 18. Trajectory of the most fit organism from the CF experimental set on a novel path. (a) The state grid has right turns only. The first directional cue is a right turn. This should be used to represent the rest of the general turn cues. It is the same situation with the right state grid which has left turns only. The green star represents the organisms starting point. The organism final position is represented by a red octagon.

Module A

```
Do
        CX = sense
        BX = BX+1
        If (BX < CX) NAND BX
        If (CX == 2) GOTO 1
        If (CX == 4) GOTO 2
        NAND BX
        If (BX > nutrient)
              1: rotate right
        move
        copy
```

```
        If (CX == 4)
              2: Go to module B
    While (CX != -1)
```

The first module is in charge of handling the right turn paths. The organism enters module A execution at least once to perform a sense instruction. After that it performs a jump depending on the sense. If the cue sensed is a right turn (value of 2), the organism jumps directly to the rotate right instruction bypassing the comparison instruction. After the first directional cue, the organism only encounters the general turn cue (value of 1) through the rest of the path. This causes the organism to skip the jump instructions because the `IF` conditions are false. The organism rotates right if the cue sensed is greater than 0 (nutrient). The organism moves and copies an instruction even if it doesn't perform a turn. The `NAND` instruction resets the BX register value, and keeps the flow of the loop. Module A exits execution when the organism steps off the path, and starts executing module B until the organism finishes replicating and then divides.

When the organism is placed on a left turn only path, the execution enters module A until the first left turn cue is found. This causes the organism to jump out of the module A execution and into the module B. The organism performs a left turn after it exits module A, but before module B starts. Once the organism enters module B execution, it is facing in the correct direction. This also means that module B only handles the general turn cue. The following pseudocode demonstrates the execution of module B.

```
    Do
        move
        BX = sense
        If (BX != empty)
```

```
            If (BX == 1) rotate left
        Copy
    While (not finished copying)
```

Module B handles the left turns for the organism. The organism moves, and performs a sense instruction. If the cue is a general turn cue, it rotates left and moves, otherwise it just moves ahead. There is no other check for the value that comes from the environment. This indicates that if organism senses anything other than the general turn cue, it moves straight ahead. Once the organism encounters an empty cell it stops moving and only continues with the offspring replication. The organism divides once replication is complete and ends execution.

From the analysis of both modules it is apparent that this organism would not be able to deal with a new environment that incorporates both right and left environmental cues. If the environment first cue is a right turn cue it can handle a left turn cue that is farther ahead. The same cannot be said when the first environmental cue is a left turn. There is nothing in module B that indicates a capability to go back and start executing module A (Figure 19). Table 6 shows the Avida instructions for the main modules described above. As mentioned previously, there is a left turn and the start of module B between both modules. The combination of both modules consists of 31 instructions. The organism is made of 148 instructions.

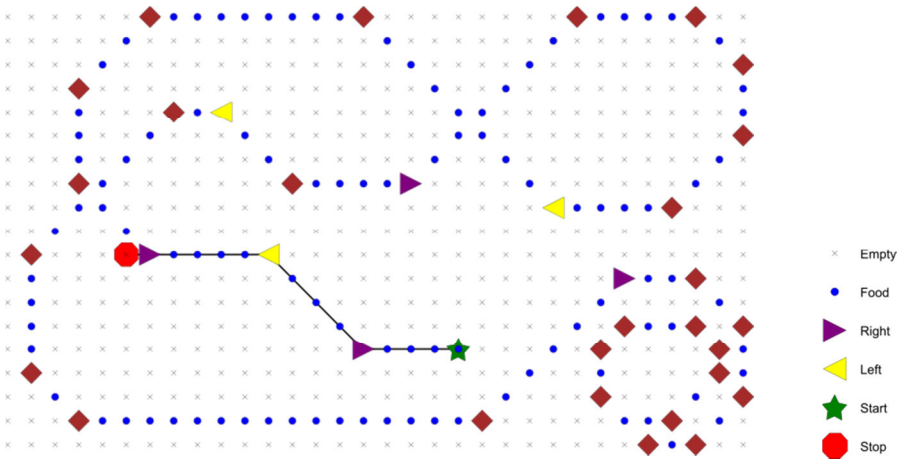


Figure 19. Trajectory of the most fit organism from the cue first experimental set on a novel path. This trajectory shows that the organisms can traverse a path where the first directional cue is a right turn (which uses module A), and once it encounters a left turn (which uses module B), it is incapable of turning right. The green star represents the organisms starting point. The organism final position is represented by a red octagon

Table 8. The Avida instructions for the CF modules. Module A handles right turns, module b handles left turns. There is a left turn and the start of module B between both modules

Module A	
Position	Instruction
76	inc
77	sg-sense
78	nop-C
79	inc
80	if-less
81	nand
82	jmp-head
83	nand
84	if-grt-0
85	sg-rotate-r
86	sg-move
87	h-copy
88	jmp-head
89	mov-head
90	if-label
91	mov-head

Module B	
Position	Instruction
95	sg-move
96	sg-sense
97	if-equ-X
98	sg-rotate-l
99	if-less
100	h-search
101	if-label
102	nop-C
103	h-divide
104	h-copy
105	shift-l
106	mov-head

General Turn Cue

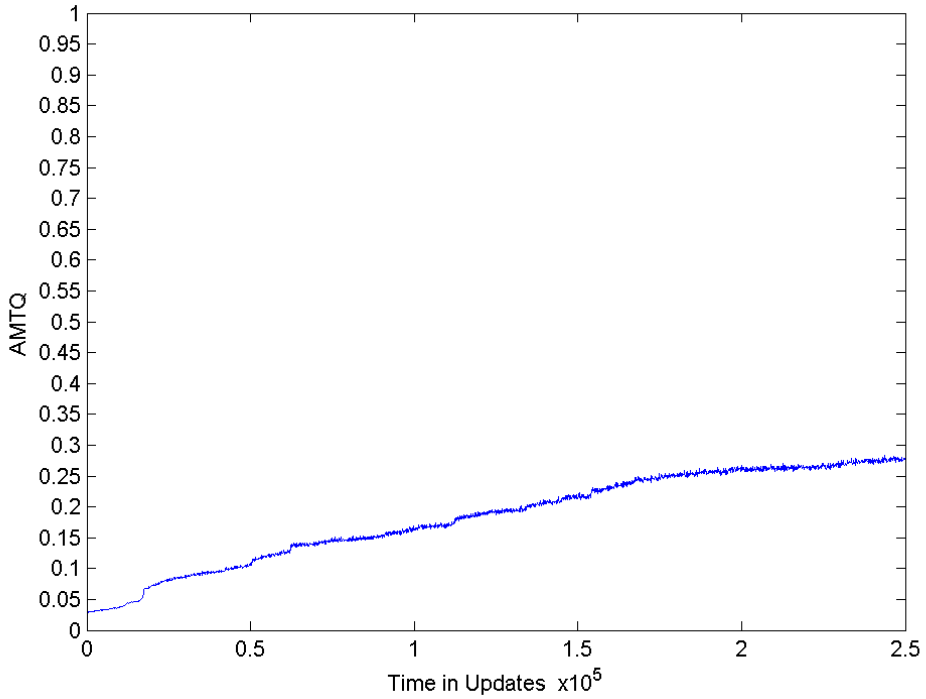


Figure 20. AMTQ for the 50 sets over time for the GT set up. This experimental set has the lowest median values compared to the all the previous common ancestor experimental set values. The organisms are traversing a little more than a quarter of the path on average.

The general turn cue (GT) environment is the most complex environmental set of this thesis. The organisms need to remember the directional cue previously encountered and repeat it. When it encounters a turn in the other direction, the "memory" needs to be updated with this new cue. This general turn cue can vary at irregular intervals on the same path, unlike the cue first experiment where the cue was the same through the path. From figure 20, we can see that, on average, the organisms are able to traverse around 25% of the path. The AMTQ distribution from figure 21 shows the median of values from the organisms is just below 0.2. This means that most populations evolve to traverse below 20% of the path, but there are a few that evolve to traverse most of the path. There is no significant statistical difference on performance by the organisms on the paths encountered during evolution (Kruskal-Wallis Test p=0.885). Table 9 shows the top five replicates from the GT experiment with their task quality (TQ).

Table 9. Top five replicates from the de novo General Turn Cue experiment with their task quality (TQ).

Replicate	TQ
25	0.985393
34	0.985393
1	0.865414
48	0.83103
11	0.750863

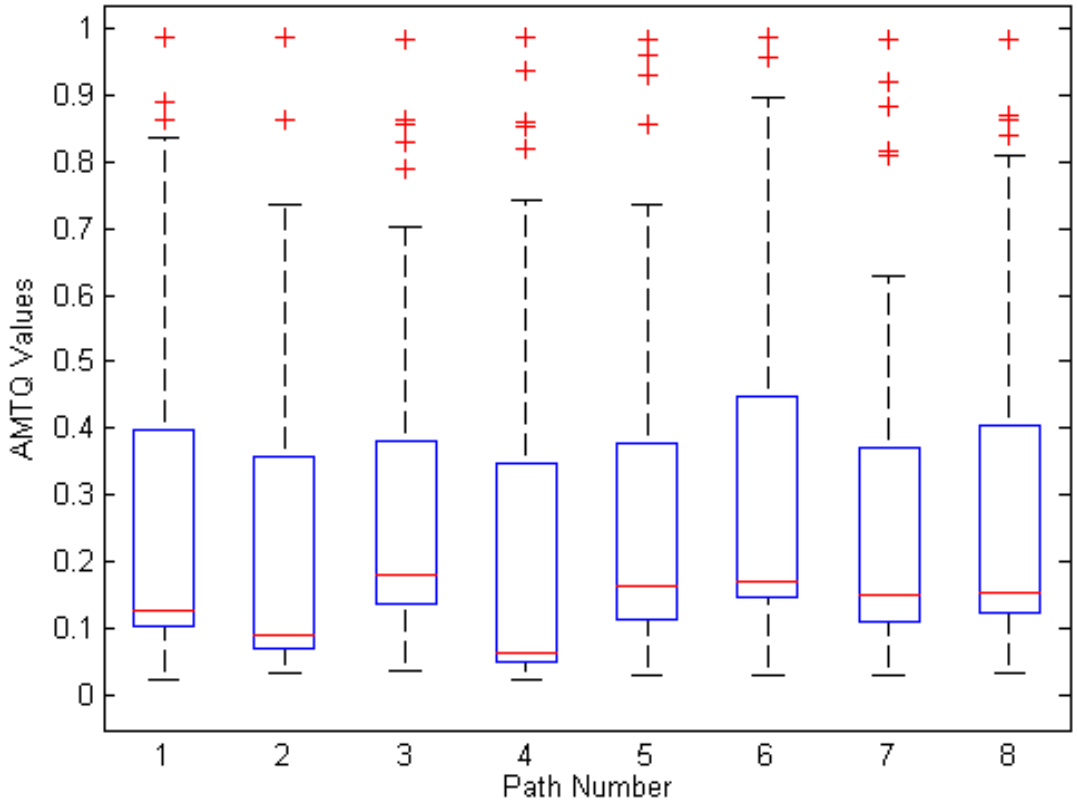


Figure 21. AMTQ value distribution for GT environment. This experimental set has a diminished median values compared to the cue first experimental set values. This means that the organisms now have difficulties evolving a solution to interpret the general turn cue.

The following is pseudocode of the organism main module that performs the path traversal task.

```
Do

        move
```

```
        CX = sense
        CX = CX * 2
        BX = BX + CX
        If (BX < CX) go to end
        Switch (CX)
        Case 'general':
             BX = Popped value
             If( BX > 0) go to right
             else go to left
        Case 'right'
             rotate right
             Push
             Break
        Case 'Left'
             rotate left
             Pop
             BX = BX - CX
             Swap BX ← → CX
             Break;
        Case 'nutrient'
             copy
             break
While ( BX > CX)
```

The organism has a single module that handles the movement and orientation of the organism in the environmental path. The organism enters the module and performs a move straight ahead. This might have evolved because none of the environments first cue is a directional cue but a nutrient. The organism performs a sense instruction and stores the result in the CX register. Then the value of the CX register gets duplicated. CX and BX are added, and the result is saved in the BX register. If BX is less than CX, the organism ends the execution module. This only happens when organism senses an empty cue. When the environmental cue is a nutrient, all the organism only copies an instruction and starts the loop again. When an organism senses a right turn cue, it rotates right, and pushes the value of BX to the top of the stack. The only push instruction in the module is performed after the organism performs a right turn. Then the organism just starts the loop execution again. In a situation where the organism encounters a left turn cue, the organism rotates left, and pops the value from the stack and saves it in the BX register. This clears the stack from the right turn value that was stored. After that, the value in BX subtracts the value of CX, and stores the result in BX. The values of BX and CX get swapped. This ensures the value going back in to the loop of BX is greater than CX, and that the loop continues. This most likely evolved this way because none of the environments end with a left turn cue. When the environmental cue is a general turn cue, the organism performs extra steps to check if the previous environmental cue was a left or right turn. The organism performs a pop, and compares the value to be greater than 0. Since the only cue that pushes a value into the stack is the right turn, if the pop has a value it turns right, otherwise it turns left. After the organism exits the loop, it continues replicating, performs a 180 degrees turn, and returns to the final grid in the state path. Figure 22 shows the organism path traversal task of the most fit organism from the cue first experiment set.

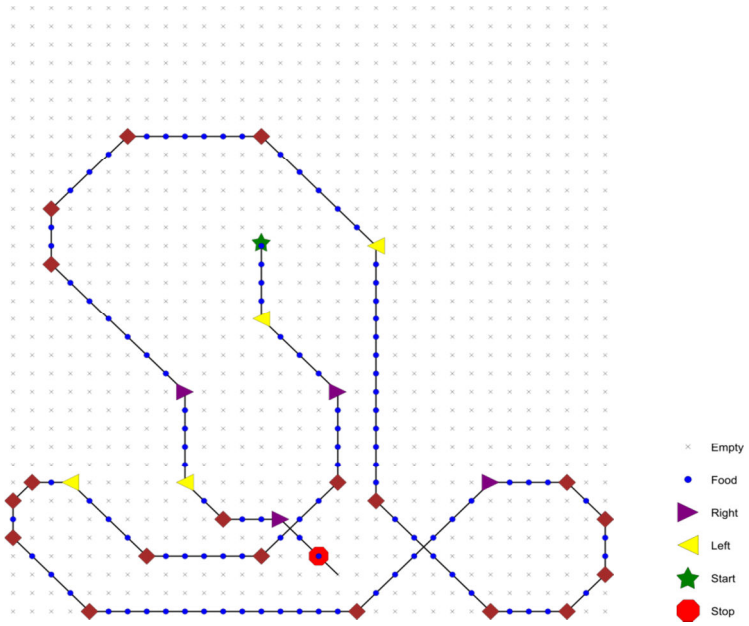


Figure 22. Trajectory of the most fit organism from the general turn cue experimental set. The organism successfully traverses the path, and returns to the last grid on the path. The green star represents the organisms starting point. The red octagon represents the organism's final position.

Table 10. Avida instructions for the main module of the top performing organisms in the GT experiments

Main Module			
Position	Instruction	Position	Instruction
35	sg-move	48	sg-rotate-r
36	sg-sense	49	push
37	nop-C	50	if-grt-0
38	shift-l	51	mov-head
39	nop-C	52	shift-l
40	add	53	shift-l
41	if-less	54	sub
42	get-head	55	sg-rotate-l
43	jmp-head	56	h-alloc
44	h-copy	57	pop
45	mov-head	58	Sub
46	pop	59	Swap
47	if-grt-0	60	mov-head

This is the environment with the most complex configuration of cues and turns, and this lead to the evolution of the most complex organism. It evolved a simple yet efficient solution to store the previous encountered directional cue, and move in that direction when a general turn cue is encountered. Table 10 shows the Avida instructions for the main module of the top performing organisms in the GT experiments. The main module is composed of 26 instructions. The final size of the organism genome is 399 instructions.

Results

Table 11 shows the AMTQ and the top performing replicate from each experimental set along with the number of replicates that have a TQ above 0.9. The experiment with the greatest AMTQ was from the RL. It is also the experiment that has the greatest number of replicates that can traverse most of the path. It seems that an organism evolves better when it encounters both right and left turn cues in the same environment. The CF and the GT had a very close AMTQ to each other, and both had 2 replicate organisms with a task quality above 0.9. The major difference is in the size of the genome. The CF organism is made of 148 instructions, while the GT organism is made of 399. Although the GT is more than double the size of the one from the CF, the number of the instructions from the module that executes the task is very close to each other. The one from the CF environment uses 31 instructions; the one from the General Turn cue uses 26 instructions. Probably the reason why there is so much difference is the use of two modules instead of one, and this adds some extra instructions to the modules.

Table 11. The AMTQ and the top performing replicate from each experimental set along with the number of replicates that have a task quality (TQ) above 0.9.

Experiment	AMTQ	Top performer Replicate	Replicate TQ	Replicates with TQ > 0.9
ST	0.526077	18	0.986024	12
RL	0.7502	5	0.992289	31
CF	0.333309	5	0.986124	2
GT	0.276561	25	0.985393	2

One Level Transplant Experiments

The purpose of the transplant experiments is to see how the previous adaptations affect the evolution of new ones. This includes the number of surviving descendants, size of the genome, size of the main module, the task quality of the descendants, and how different ancestor adaptations affect the evolution of the descendants. The first set of transplant experiments is the one level transplant. Here the top five ranking organisms from each environment that evolved *de novo* are used to seed the next more complex environment where additional evolution occurs. The second set of transplant experiments is the waterfall transplant. The top five performing organisms get transferred from ST all the way through the GT. I conducted 10 replicate experiments for each seed organism, for a total of 50 experiments. Because of the differences between the environment of evolution and the transplant environment, some populations did not survive to the end of the transplant experiment. Therefore, the AMTQ includes only the populations that survived the transfer to the new environments. For example, if of the 50 experimental populations only 30 survive, the AMTQ would be calculated on those 30 experiments.

The one level transplant experiments are as follows: from Single turn to the Right and Left turns environment (ST-RL), from the Right and Left turns environment to the Cue First

environment (RL-CF), from the Right and Left turns environment to the General Turn Cue environment (RL-GT), and finally from the Cue First environment to the General Turn Cue environment (CF-GT). The path trace will not be included in this section because it has been demonstrated in the previous section that the top performing organisms are able to successfully traverse the paths.

Single Turn to Right and Left

The single turn to right and left (ST-RL) is the first transplant experiment. The top five ranking organisms from ST are transplanted to the RL environments. Since the top performing organism evolved capabilities to traverse a path with both right and left turns in the ST environment, the organisms should be able to evolve a solution faster that the evolving de novo. Figure 23 shows that this is, indeed the case. The organisms evolved the capability to traverse the paths with both right and left turn cues after 300 updates approximately compared to the solution evolved by the de novo experiments which are only able to traverse 75% of the path by the end of evolution. Further analyses of the results show that 10 of the 50 replicate experiments were not able to evolve a solution, and actually died before the 200^{th} update. All of these 10 experiments that did not survive had the fourth ranking organism as a common ancestor. This indicates that the solution evolved by this organism from the ST was not as evolvable as the other transplant seeds. The top ranking organism from the ST-RL came from the third ranking organism and not from the top performing organism from the ST. This might be because the top performing organism already had the capability to turn both right and left that additional evolution did not provide much of a fitness improvement over the other seed organisms. Table

12 shows the top five ranking organisms from the ST-RL experiment, their task quality (TQ), and the seed replicate they come from.

Table 12. Top five organisms from the ST-RL experiment, their task quality (TQ), and the seed replicate they come from.

Replicate	TQ	Seed
29	0.993195	3
23	0.992289	3
26	0.992289	3
27	0.992289	3
24	0.985725	3

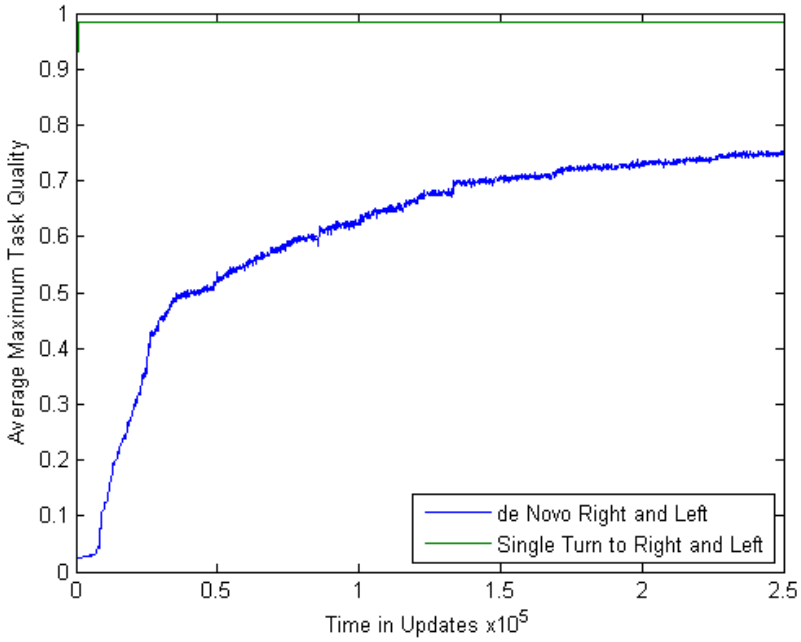


Figure 23. The average maximum task quality (AMTQ) for the RL. The curve shows that the transplanted organisms perform better that the ones that evolved de novo. The transplanted organisms can traverse most of the path by the end of evolution.

The following is the pseudocode from the main module that handles the path traversal task from the fittest organism in the ST-RL experiment.

```
Do
    If (BX != -1)
        rotate right
```

```
            move
            BX = sense
        rotate left
        If (BX > right turn) rotate left
        BX = BX / 2
        If (BX == 1) rotate right
        Copy
        copy
    while(not finished copying)
```

The solution evolved by the organism is similar to the one that evolved in the ST environment, where the organism needs to correct the orientation once it starts the execution of the module. When the organism enters the main module it rotates right, going back to the original facing, and then moves straight ahead. Then the organism will execute a sense instruction and turn left. This turn will offset the rotate right instruction at the beginning of the module if the organism does not encounter a rotate cue. If the value sensed is greater than a right turn (*i.e.*, if greater than 2), it will turn left. After that it will divide the cue sensed by two, and compare the result to 1. If the result is equal to 1, the organism will perform a right turn. After that it will copy two of its instructions to the genome for the offspring, and start the main module execution again. Once the organism steps off the path it will continue rotating in place and copying the genome until it finishes copying the genome. Once the genome is copied, the organism will exit the main module, and continue executing instructions outside of the main module until it divides. Table 13 shows the Avida instructions for the main module for the fittest organisms in the ST-RL experiments. The final genome size of this organism is 467 instructions, and the main module is made of 21 instructions. Figure 24 shows the AMTQ values distribution

from the default ancestor. Right and Left Turns experiment and from the Right and Left Turns experiment with the organisms transplanted from the Single Turn experiment. Similar to the plot from figure 24, almost all of the values have an AMTQ very close to 1. There is a significant difference in the performance of both RL and ST-RL performance (Mann-Whitney U-Test $p=1.53x10^{-9}$).

Table 13. Avida instructions for the main module for the fittest organisms in the ST-RL experiments

Main Module			
Position	Instruction	Position	Instruction
185	sg-rotate-r	196	sg-rotate-r
186	sg-move	197	if-label
187	sg-sense	198	h-copy
188	if-less	199	h-copy
189	h-search	200	sg-sense
190	sg-rotate-l	201	if-label
191	if-grt-X	202	nop-C
192	nop-B	203	nop-A
193	sg-rotate-l	204	if-grt-X
194	shift-r	205	mov-head
195	if-equ-X		

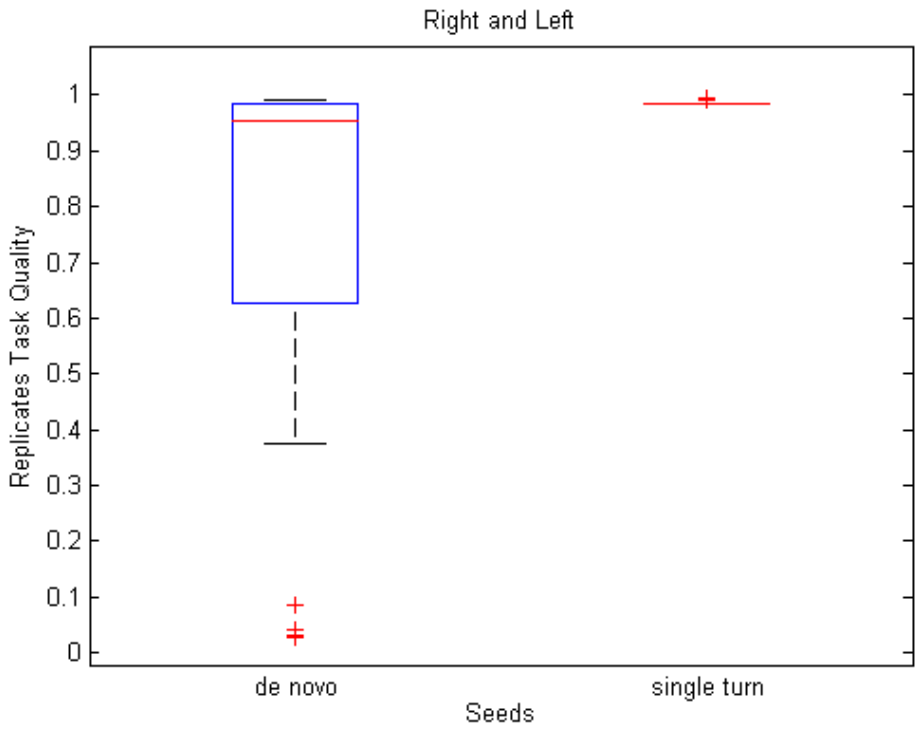


Figure 24. AMTQ values distribution from the de RL experiment and the ST-RL experiment.

Right and Left to Cue First

The second transplant experiment is from the Right and left environment to the cue first environment (RL-CF). For this experimental set up, only two of the five organisms transplanted were able to evolve a solution. This means that only 20 experiments of 50 survived all of the 250,000 updates. The fittest organism from the previous environment was not able to survive. All of the surviving organisms come from the third and fifth ranked organisms that were transplanted from the RL set. Table 14 shows the top five performing replicates from the RL-CF, their task quality (TQ) and the rank of its transplanted ancestor.

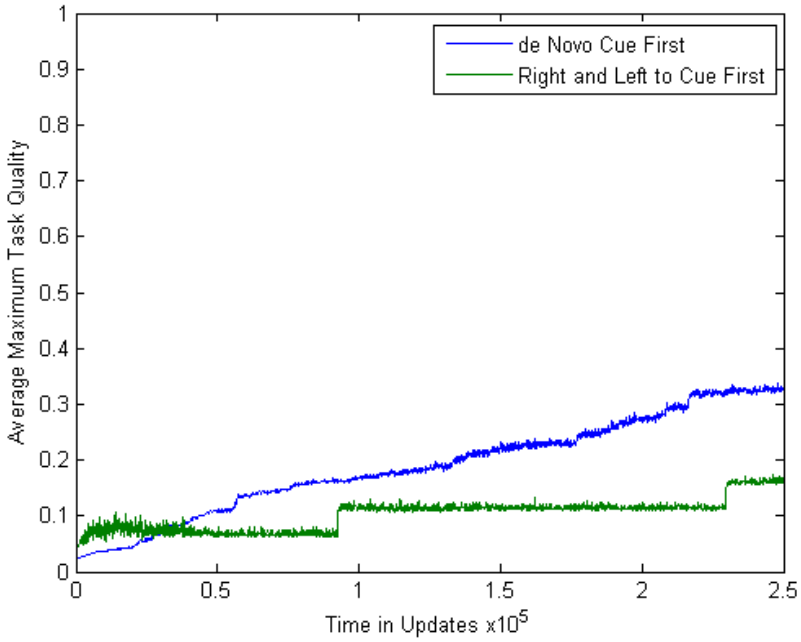


Figure 25. The average maximum task quality (AMTQ) for the Cue First experiments. The curve shows that the transplanted organisms do not perform better than the one that evolved de novo.

Figure 25 shows that the AMTQ of the transplanted organisms was better at the beginning of the experiments. This is most likely due because the organisms are already capable of traversing at least part of the path. As evolution progressed, the AMTQ of the *de novo* experiments passed that of the transplant. This happens around update 350. After further analysis, the third ranked organism from the RL generated an AMTQ that was close to 0.4 at the beginning, but as evolution progressed its AMTQ started decreasing. This phenomenon accounts for the noise in the first thousand updates in the plot. The fifth ranked organism generated a solution similar to the default ancestor experiment, but the AMTQ specific to that transplanted organism was not as good as the AMTQ of the CF although the performance from all of the surviving experiments is half of the one that evolved *de novo*, the most fit organism has a greater AMTQ. The task quality of the top performing organism in the CF population is 0.986124, while the one from the transplanted experiment from the right and left turns experiment has a task

quality of 0.988801. Only two of the twenty replicates that survived the whole experiment were able to successfully traverse most of the cue first environment paths. Both of these organisms evolved from the fifth ranked transplanted ancestor.

Table 14. Top five performing replicates from the RL-CF, their task quality (TQ) and the rank of its transplanted ancestor.

Replicate	TQ	Seed
48	0.988801	5
50	0.986124	5
25	0.084	3
29	0.083853	3
43	0.080308	5

An analysis of the trace from the organism shows that it only uses one main module to replicate and move through the state grid. The fittest organism from the CF used two modules, and could not successfully traverse a path when both left and right directional cues were present in the environment. Since the current organism only uses one module, it can successfully traverse paths with right, left, and general turn cues. This means that this organism would be able to survive in the GT environment (Figure 26). The following is the pseudocode from the main module of the fittest RL-CF organism.

```
Do
     If( BX > 1) Push BX to Stack
     BX = BX / 2
     If(BX > 1) rotate left
     move
     BX = sense
     If(BX == General turn cue) BX = pop value from stack
     IF(BX == Right Turn) rotate right
```

```
        If (finished copying) divide
        copy
    While (not divide)
```

This organism evolved a more straight forward solution for traversing the path. The first instruction in the module compares the BX register value to be greater than 1. The BX value is only greater than 1 when the cue sensed is either a left turn (value of 4) or a right turn (value of 2). When this condition is true, the value is pushed onto the organism's stack, saving the type of turn it will use when a general turn cue is sensed. After that the value in the BX register is divided by 2, and if the value is greater than 1, the organism it will turn left. The organism moves to the next grid cell that is facing, and performs another sense instruction. When the general turn cue is sensed, the instruction saved in the stack gets popped to the BX register. This action serves to map the previously encountered turn direction (right or left) to the current general turn cue. If the value stored in the BX registered is equal to right turn (value of 2), the organism will make a right turn and copy an instruction for its offspring. The module will stop execution once the organism finishes replicating. Unlike the other organisms previously analyzed, this organism keeps traversing the path until it finishes replication. It does not stop moving once the organism senses an empty cell. This does not strongly affect the task quality of the organism. Its only risk is running out of energy from the execution of the extra instructions. Table 15 shows the Avida instructions for the main module for the fittest organisms in the RL-CF experiments. The top performing organism has a genome size of size of 136 instructions, and the main module is comprised of 25 instructions. Figure 27 shows the TQ distribution from the CF experiment and the RL-CF. There is a significant statistical difference in the performance of both CF and RL-CF performance (Kruskal-Wallis Test p=0.0003).

Table 15. Avida instructions for the main module for the fittest organisms in the RL-CF experiments.

Main Module			
Position	Instruction	Position	Instruction
107	if-grt-X	120	sg-rotate-r
108	push	121	if-label
109	shift-r	122	nop-C
110	get-head	123	nop-A
111	if-grt-X	124	h-divide
112	sg-rotate-l	125	h-alloc
113	sg-move	126	if-grt-X
114	sg-sense	127	if-label
115	if-equ-X	128	h-copy
116	pop	129	if-grt-X
117	get-head	130	if-grt-0
118	if-equ-X	131	mov-head
119	nop-B		

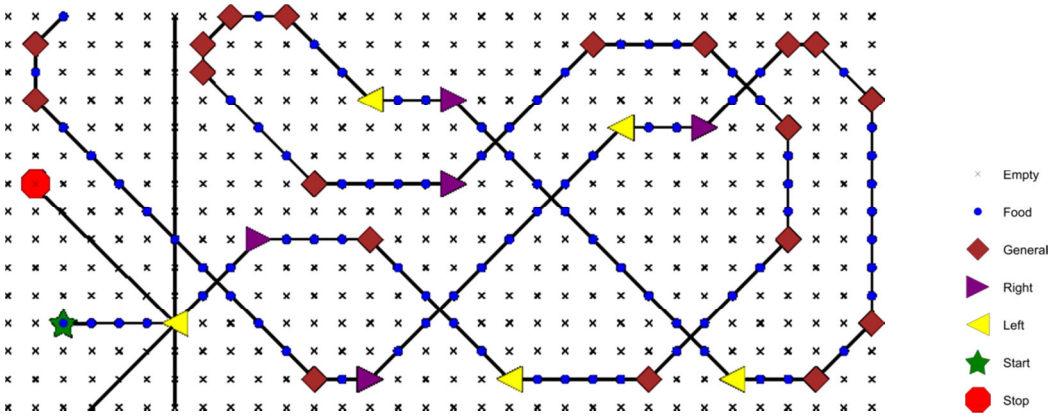


Figure 26. Trajectory of the most fit organism from the cue first environment with organisms seeded from the right left cues environment. The organism successfully traverses the path and keeps moving even after encountering an empty cell. The green star represents the organisms starting point. The red octagon represents the organism final position.

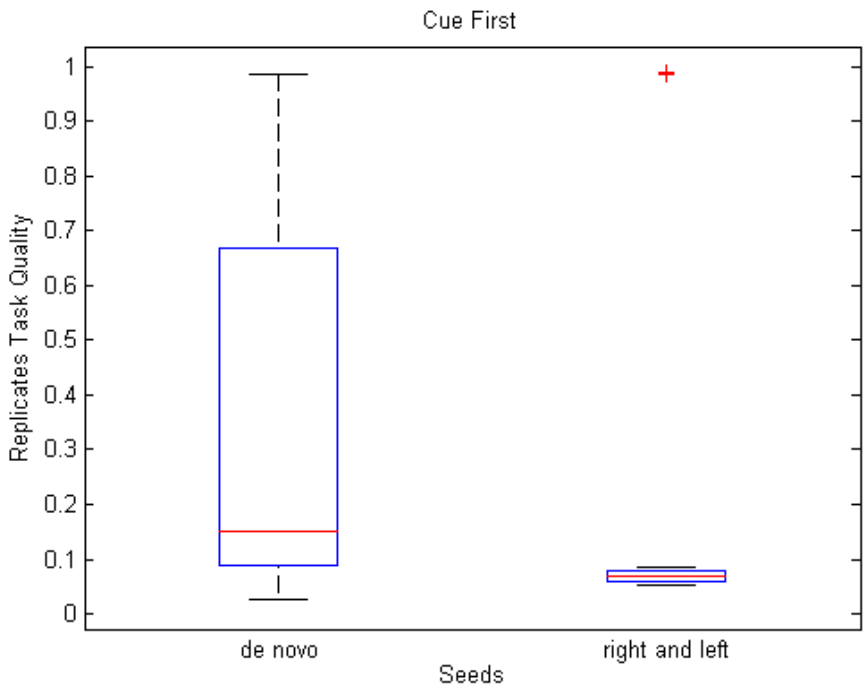


Figure 27. Task Quality distribution from the de CF and the RL-CF.

Right Left to General turn cue

The third transplant experiment is the right and left to the general turn cue (RL-GT). The result from the transplanted organisms is very similar to the one that was transplanted to the Cue first experiments; only 20 of the 50 organisms survived the experiment. 10 were from the third ranked transplanted organism, and the other 10 were from the one ranked fifth. Figure 28 shows that the AMTQ of the GT and the one RL-GT experiments are very close to each other. At the beginning, the transplanted organisms perform better. This may be because the transplanted organisms have the capability to move and follow turns to both right and left. As the organisms evolved, the *de novo* organisms surpass the AMTQ of the transplanted organism. The transplanted organisms appear to evolve some adaptations to the new environments around update 9,000 update causing the AMTQ values to jump, close to the organisms that evolved *de novo*. The organisms that evolved from scratch steadily increase performance over time, but are

surpassed by the transplanted organisms at update 249,000, just before the experiment finishes. The organism with the greatest task quality was from RL-GT experiment, and had an AMTQ of 0.9926. Table 16 shows the top five performing replicates from the RL-GT experiment, the task quality of the replicate (TQ), and rank of its ancestor.

Table 16. Top five performing replicates from the RL-GL, their task quality (TQ) and the rank of its ancestor.

Replicate	TQ	Seed
21	0.992695	3
41	0.985393	5
44	0.985393	5
47	0.962569	5
25	0.332553	3

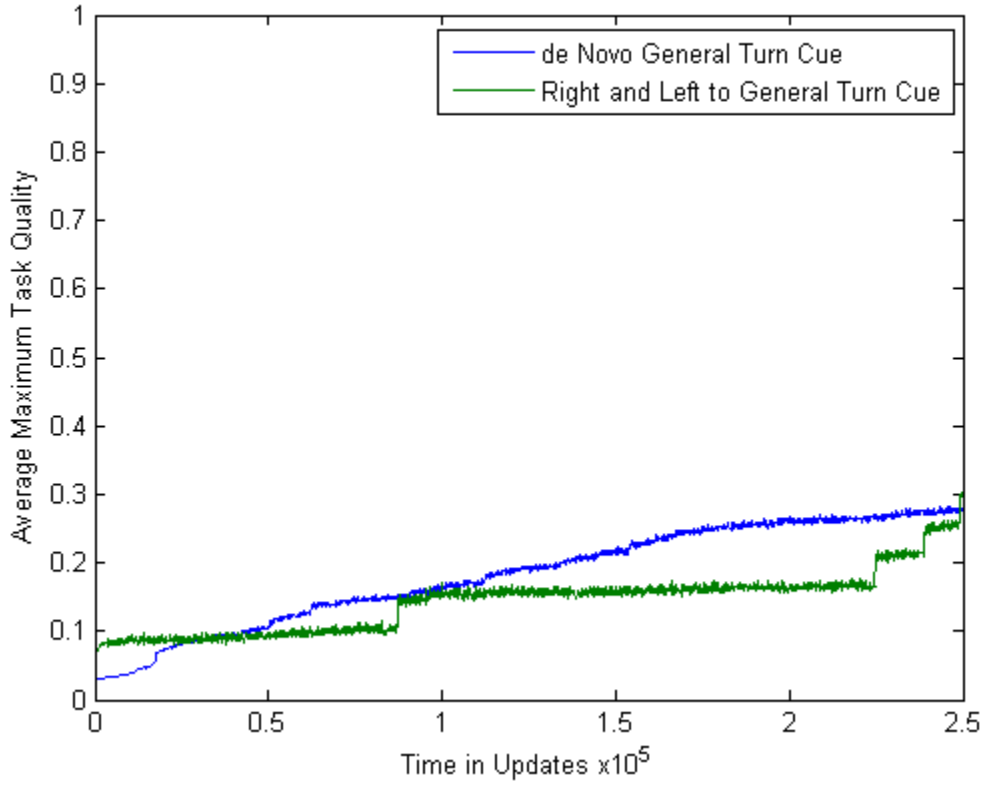


Figure 28. The average maximum task quality (AMTQ) for GT. The curve shows that the transplanted organisms are able to perform better than the ones that evolved de novo.

The following is the pseudocode of the most dominant organism in the RL-GT

```
Do
If(BX != -1)
      If(BX > 0) CX = sense
      If(BX > 1) CX = Instruction Pointer
      move
      BX = sense
      CX = BX + CX
      IF(BX == 1) swap BX and CX
      BX = BX /2
      If(BX > 1) rotate left
      If(BX == 1) rotate right
while (not finished copying)
```

The fittest organism from this experiment evolved a simpler solution than the one that evolved from the simple common ancestor. This solution does not involve pushing information to the stack, and fewer instructions are used for path traversal, just two fewer instructions than the algorithm that evolved from the default ancestor.

The first two `IF` statements are used as a control for the general turn cue. If a right turn cue is sensed the organism stores the value in the CX register. If the cue sensed is a left turn, the organism copies the current value of the instruction pointer in the CX register. The organism then moves in the direction it is facing, senses the next environmental cue, and stores the sensed value in the BX register. The organism adds the values of the BX and CX register, and stores the result in the CX register. A comparison is made to see if the value in the BX register is equal to the general turn cue (value of 1); if this condition is true the values in the CX and BX registers

will be swapped. The value in the BX register is then divided by two and the result value is compared to be either greater than or equal to one. When the value in BX is greater than one the organism performs a left turn; when the value is equal to one, the organism performs a right turn. When the cue sensed is an actual left or right turn, the swap between the values of the BX and CX registers is not done, while it is done when the cue sensed is a general turn. Finally the organism copies an instruction for its child genome, and starts the module's loop to continue traversing the path. The organism stops moving when it senses an empty cue, and the organism will stop execution when it divides the offspring genome. Table 17 shows the Avida instructions for the main module of the fittest organism in the RL-GT experiment. The organism is made of 211 instructions, and the main module is comprised of 24 instructions. Figure 29 shows the TQ distribution from GT experiment and the RL-GT experiment. The difference in performance from both experiments is not significantly different (Mann-Whitney U-test, $p = 0.4164$)

Table 17. Avida instructions for the main module for the fittest organisms in the RL-GT experiments

Main Module			
Position	Instruction	Position	Instruction
186	if-grt-0	198	if-grt-X
187	sg-sense	199	sg-rotate-l
188	nop-C	200	if-equ-X
189	if-grt-X	201	nop-A
190	get-head	202	h-search
191	sg-move	203	if-equ-X
192	sg-sense	204	sg-rotate-r
193	add	205	if-label
194	nop-C	206	nop-A
195	if-equ-X	207	h-divide
196	swap	208	h-copy
197	shift-r	209	mov-head

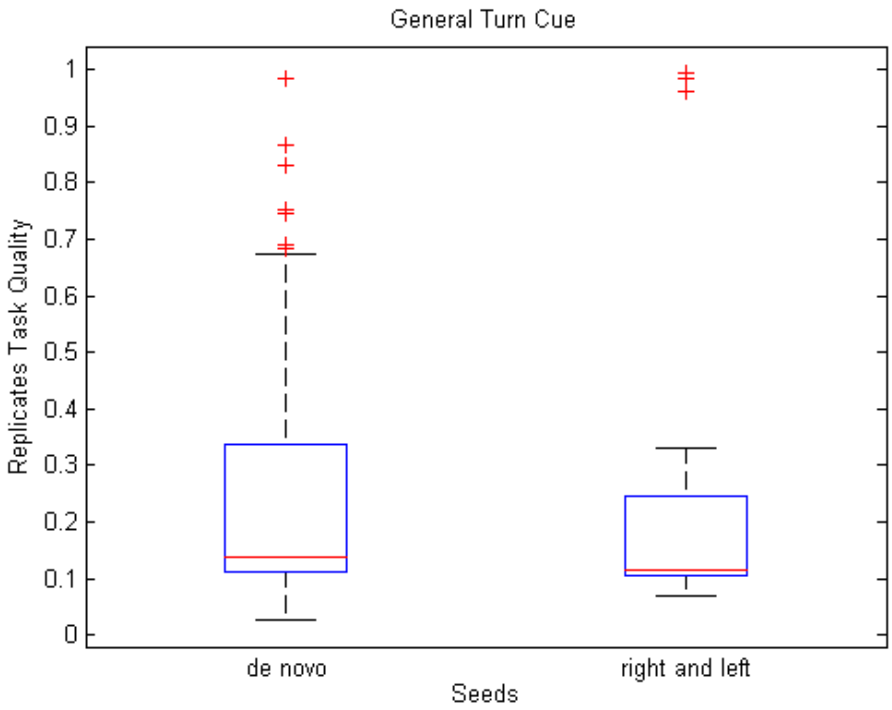


Figure 29. The TQ distribution from the GT and the RL-GT experiment.

Cue first to General turn cue

The fourth transplanted experiment is from the cue first to the general turn cue (CF-GT). The populations evolved from the transplanted organisms from the CF performed better than those transplanted from the RL. Twenty-eight of the fifty organisms survived the experiment; this is a noticeable improvement over the RL transplants. The organisms that survived the experiment came from the first, second and fourth ranked organism.

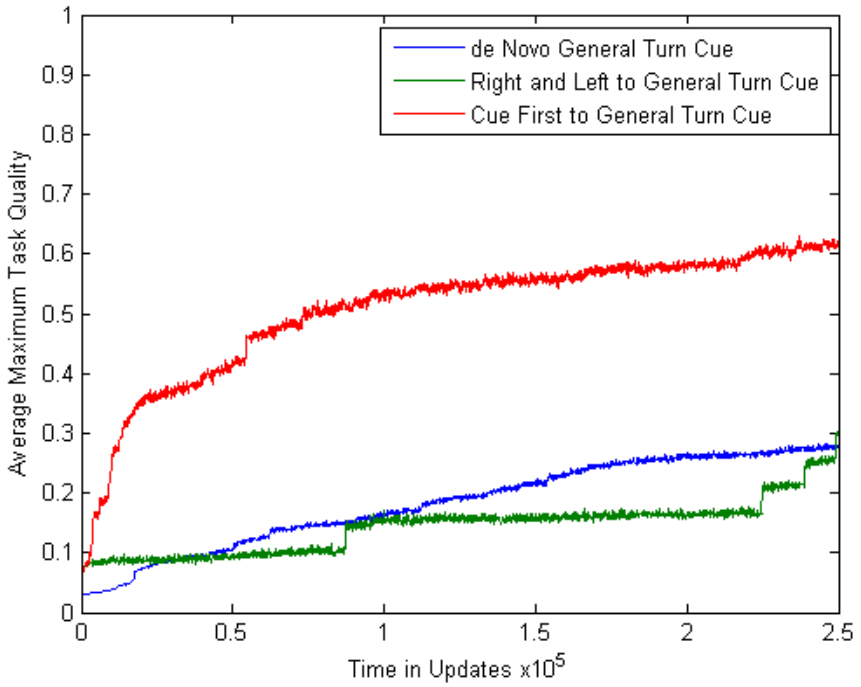


Figure 30. The average maximum task quality (AMTQ) for General Turn Cue experiments. The curve shows that the transplanted organisms from the Cue First experiments are able to perform better than the ones that evolved de novo and from the Right and Left turn seeds.

Figure 30 shows that the CF-GT was the best AMTQ of the GT experiments so far. The organisms started performing better than the ones from other experiments very quickly. The organisms reached an AMTQ of 0.5 before update 100,000, and after that time the AMTQ increases steadily. By the end of the experimental run the organisms had an AMTQ greater than 0.6. This increment in performance might be because it is more complicated to evolve an effective way to handle the general turn cue, than to turn right or left when a turn cue is encountered. The most fit organisms from this experimental set has an average fitness of 0.985393, while the most fit organism from the experimental set for the right and left seeds has an average fitness of 0.992695. Even when the top organism has a smaller fitness, the whole population set performed better than those of the other treatments; more organisms are able to survive, and they have a better overall performance. Table 18 shows the top five performing

replicates from the CF-GT experiment, the task quality of the replicate (TQ), and rank of its ancestor.

Table 18. Top five performing replicates from the CF-GT experiment, the task quality of the replicate (TQ), and rank of its ancestor.

Replicate	TQ	Seed
18	0.985393	2
32	0.983568	4
5	0.980766	1
38	0.968479	4
34	0.963104	4

The following is the pseudocode for the fittest organism from CF-GT. It is made of two main modules that handle the path traversal and replication of the organism. Module A primarily handles the right rotation and its corresponding general turn cue. Module B handles the left rotation, its corresponding general turn cue, and replication of the organism.

Module A.

```
Do
     move
     BX = sense
     If(BX != nutrient) rotate right
     BX = BX / 2
     If(BX != 0 || BX != 1)
          Rotate left
          Rotate left
          copy
          copy
          copy
```

```
            copy
            break
    While()
```

Module A handles the right turns in the state grid. Once this module starts execution it moves to the next cell that is facing. Then the organism performs a sense of the current cell, and stores the sensed value in the BX register. If the cue sensed is anything other than a nutrient, the organism performs a right turn. Then the value that is stored in the BX register is divided by two, and is used to perform the next comparison operations. The organism already performed a right turn, so the organism checks to see if the value sensed is not a general turn cue or a right turn. This indicates that the cue sensed is a left turn, and the organism rotates left two times to maintain the correct facing to traverse the path. After that the organism copies four instructions and exits module A execution. If the cue sensed is not a left turn, the execution just goes back to the beginning of the module and starts again. This module only copies instructions for replication just before it exits its execution.

Module B.

```
Do
      BX = sense
      If(BX != 0) BX = BX / 2;
      If(BX == 1)
            Rotate right
            go to Module A
      If( BX != -1) Move
      BX = sense
      If( BX == 1) Rotate left
```

```
        copy
        If(BX < CX) divide
    While(BX < CX)
```

Module B handles the left turns in the state grid. When the organism enters the module B it is already facing in the correct direction. Then the organism senses the current cell, and stores the value sensed in the BX register. If the cue sensed is anything other than a nutrient (value of 0), the sensed value will be divided by two. A sensed value equal to one indicates that the cue is a right turn. The organism would rotate right, and then it would end module B execution, returning execution to module A. Then a comparison is made to check if the value sensed is not empty. The organism moves ahead when the value sensed is not empty. After that the organism senses the environment again, and checks if the value is a general turn cue. When it is a general turn cue the organism makes a left turn. Before the module ends execution, the organism copies one instruction, and checks if it is still on the path (*i.e.* the sensed value is not the empty cue). Encountering an empty cue causes the organism to stop moving, while it continues to copy and tries to divide. The organism will stop execution once it successfully replicates. Table 19 shows the Avida instructions for the modules A and B of the fittest organism in the CF-GT experiment. This organism used the greatest number of instructions to handle the path traversal tasks. The combination of both modules A and B has 72 instructions; this includes instructions between modules that are executed when transitioning from one to the next. The whole organism genome is made of 344 instructions. Figure 31 shows the TQ distribution from the GT, the RL-GT, and CF-GT experiments. There is a significant statistical difference between the CF-GT and the other two groups, GT and RL-GT (Kruskal-Wallis, $p=3.54 \times 10^{-5}$). Figure 32 shows the Bonferroni adjustment to do multiple comparisons, and it shows that statistical differences between the groups.

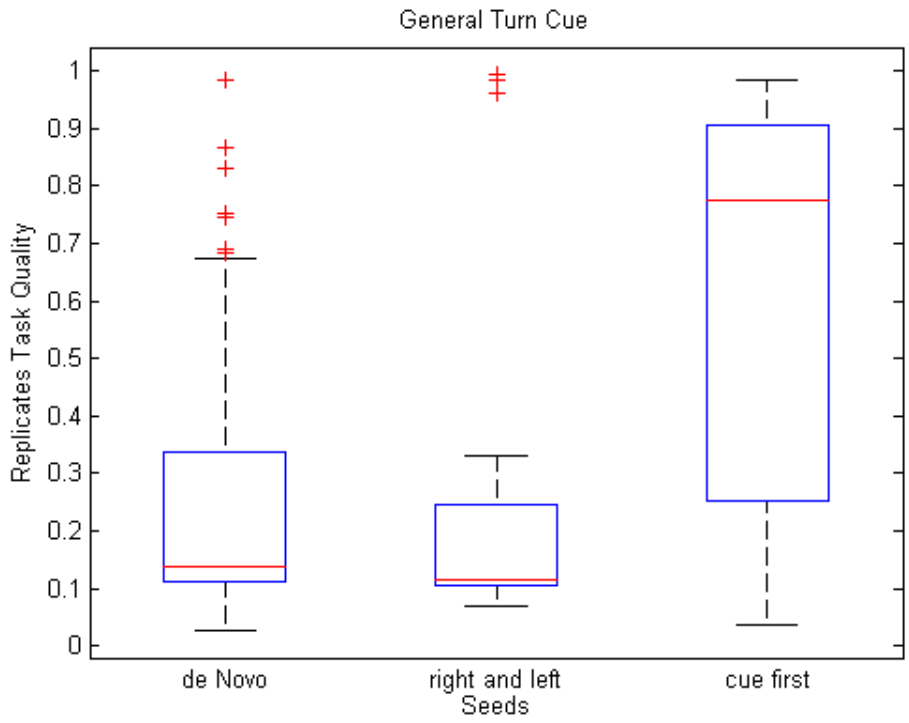


Figure 31. The TQ distribution from the GT, RL-GT, and CF-GT experiments.

Table 19. Avida instructions for the modules A and B of the fittest organism in the CF-GT experiment.

Module A	
Position	Instruction
204	sg-move
205	sg-sense
206	if-n-equ
207	sg-rotate-r
208	shift-r
209	if-n-equ
210	if-equ-X
211	mov-head
212	if-grt-0
213	pop
214	inc
215	swap-stk
216	h-copy
217	sg-rotate-l
218	h-copy
219	h-search
220	nand
221	nop-C
222	h-copy
223	swap-stk
224	h-search
225	h-copy
226	if-label
227	nop-C
228	nop-B
229	if-n-equ
230	mov-head

Module B	
Position	Instruction
240	sg-sense
241	if-n-equ
242	shift-r
243	if-equ-X
244	sg-rotate-r
245	if-n-equ
246	swap-stk
247	if-equ-X
248	h-search
249	nop-A
250	Inc
251	if-grt-0
252	sg-move
253	sg-sense
254	if-equ-X
255	sg-rotate-l
256	Add
257	nop-B
258	h-copy
259	if-grt-0
260	nop-A
261	Push
262	sg-sense
263	if-less
264	nop-B
265	h-divide
266	if-n-equ
267	shift-r
268	mov-head

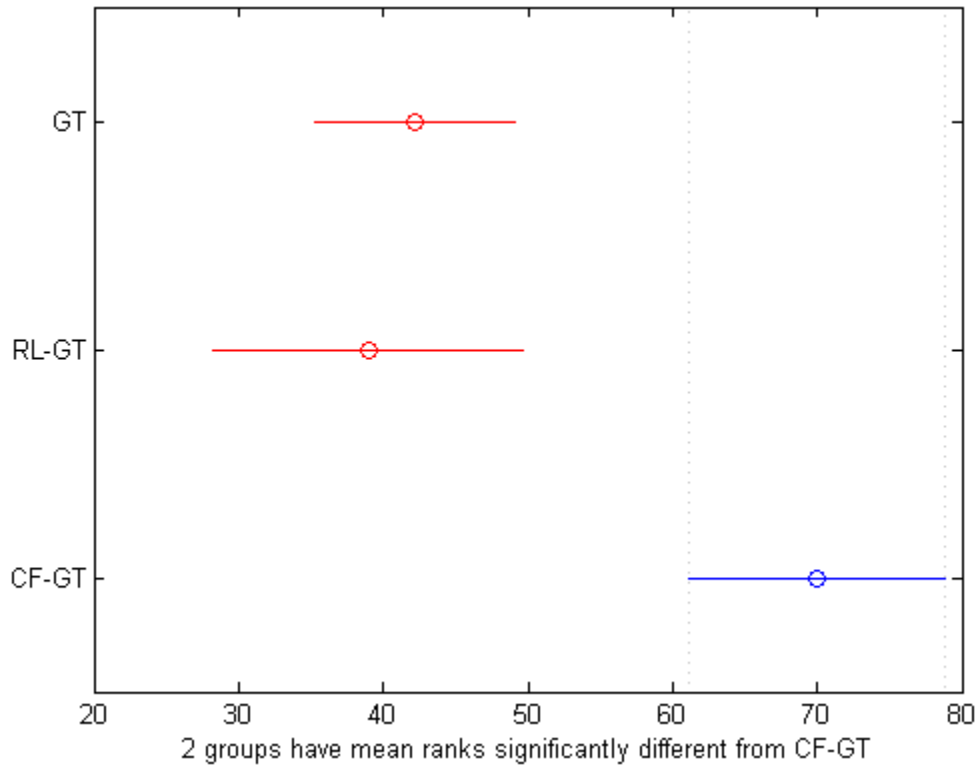


Figure 32. Bonferroni adjustment for multiple comparison tests based for the GT, RL-GT, and CF-GT showing that CF-GT is significantly different from the other GT experiments.

Results

Table 20 shows a summary of the one level transplant experiments, the number replicates that survived the transplant and the number of replicates that had a task quality (TQ) above 0.9. None of the top performing organisms from each experiment came from the top performing organism on the previous experiment. This might indicate that once an organism succeeds in an environment it is difficult for it produce the changes needed to produce the most successful organisms in another more complex one. In some of the cases, the top organism was not able to produce successful descendants that could survive in the new environment. The ST-RL top ranked organisms had 467 instructions; the number of instructions has almost doubled. Still the main module is comprised of 21 instructions, just 3 more than the RL experiment. The CF top organism had 136 instructions. These are fewer instructions than the RL-CF experiment. The

main module has 25 instructions; fewer compared to the 31 from the common ancestor organisms. The top ranked organisms from the RL-GT had fewer instructions, almost half, than the GT experiment. The main module is roughly the same size. The top performing organisms for the CF-GT also had a smaller genome (344 instructions), but 72 of the instructions compose both modules. The difference in size of the main modules is because instead of evolving a solution, that encompasses both turns and rewriting the main module, it just adapted a way to jump back to the first module. This is more than double the instructions for both, the CF and RL-CF environments.

The genome size decreases for organisms that are evolving successful adaptations, but once a successful adaptation appears the size of the genome starts increasing. Most of the organisms survived a new environment when a complex adaptation was already present, and it only needs to be adjusted for the new environment. This is most notable on the ST-RL and the CF-GT transplant experiments. In both cases the genome size of the organisms increased, the number of surviving replicates increased, and the replicates with an AMTQ greater than 0.9 increased.

Table 20. The one level transplant organism experiment, the top performing replicate, the rank of the seed organism, the number of replicates that survived the transplant, and the number of replicates that had an AMTQ greater than 0.9.

Experiment	Top Replicate	Seed Rank	Surviving Replicates	Replicates with AMTQ > 0.9
ST-RL	26	3	40	40
RL-CF	48	5	20	2
RL-GT	21	3	20	4
CF-GT	18	2	28	7

Waterfall Transplant Experiments

The waterfall transplant experiments are the last experimental treatment for this thesis. The top five organisms from the ST experiment are used to seed the RL environments. Then the top five organisms resulting from this seeded experimental set are used to seed the CF experiments and the GT directly. Finally the top five organisms from the seeded CF are transplanted to the GT. The results of the one level transplant experiments demonstrated that most of the transplanted organisms were not able to survive the transfer from the right left environment to the cue first environment, but populations seeded with two transplanted organisms performed better than the populations evolved de novo. It was also observed that the organisms from the CF-GT performed better than the RL-GT.

General Turn from Right and Left Waterfall Transplants

The first waterfall transplant experiment is the general turn from right and left waterfall experiments (GT from RL waterfall). The GT from RL waterfall transplant is the experiment with the lowest AMTQ among all the experimental treatments at this point. All of the 50 populations survived the experiment. The AMTQ for the experiments can be seen in figure 33. All the experiments from the set survived experiment execution. During the first half of the experiment, the performance of the organisms is very similar to the performance of the populations from RL-GT environment. After this tie point, evolution appears to level off, and in the end, the experiment has the lowest AMTQ among all the experimental treatments. Table 21 shows the top five performing replicates from the GT from RL waterfall experiment.

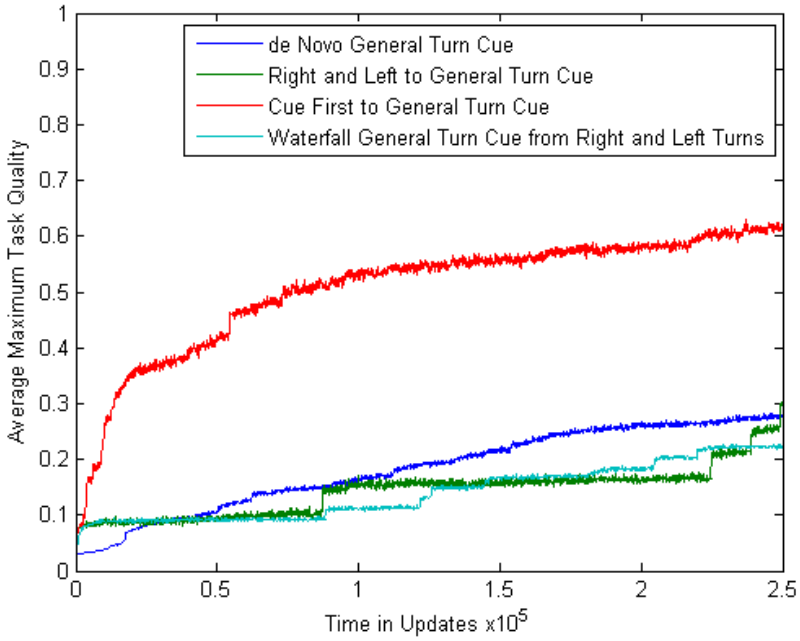


Figure 33. The average maximum task quality (AMTQ) for General Turn Cue experiments. The curve shows that the transplanted CF-GT organism has the best AMTQ, while the GT from RL waterfall has the lowest AMTQ form all the experiments.

Table 21. The top five performing replicates from the GT from RL Waterfall experiment.

Replicate	TQ	Seed
27	0.988835	3
43	0.986261	4
1	0.985393	1
16	0.979648	2
18	0.972098	2

The following is the pseudocode for the top performing organisms from the GT from RL waterfall

```
Do
    IF (BX < CX)
        BX = BX*2
        IF (BX > 0) Push BX to Stack
```

```
            rotate right
            move
        BX = sense
        IF (BX = 1) Pop BX from stack
        copy
        rotate left
        IF (BX > 2) rotate left
        copy
        BX = BX / 2
        IF (BX == 1) rotate right
        copy
        CX = sense
    while (not end label)
```

This organism evolved one module that is in charge of traversing the path, and copying instructions for its offspring. When the organism enters the module it has already left turn. The first instruction in the module checks if the value in the BX register is less than the one in the CX register. This test is true only when the organism senses an empty cue. If the value of the BX register is greater than 0 (nutrient) then the organism will push the value in the BX register onto the stack. This action will save the directional cue encountered. Then the organism makes a right rotation and moves to the next cell. Then the organism executes a sense instruction and stores the sensed value in the BX register. If the cue sensed is equal to one (general turn cue), the value stored in the stack gets popped, and the value is stored in the BX register; then the organism knows which turn was the previously encountered. The organisms will copy an instruction and rotate left. When the value stored in BX is greater than two (left turn cue), the

organism makes a left rotation. The additional left turn results in the organism's facing the correct direction after encountering the left turn cue. Then BX register value gets halved and compared to the value one. If the comparison is true, the organism turns right. Then the environment is sensed again before moving, but this time the value is stored in the CX register. This stored value will be used as control for when the organism senses an empty cue at the beginning of the module. When the organism senses an empty cell, the organism will stay in place, rotate, and copy an instruction for its offspring. This module does not handle the organism's division; instead the organism will keep executing, and divide later in the genome. The main module for the organism is made of 28 instructions. Table 22 shows the Avida instructions for the main module. The organism's genome is composed of 476 instructions.

Table 22. Avida instructions for the main module of the fittest organism in the GT from RL waterfall experiment.

Main Module			
Position	Instruction	Position	Instruction
1	shift-l	15	nop-B
2	h-alloc	16	sg-rotate-l
3	if-grt-0	17	h-copy
4	push	18	shift-r
5	sg-rotate-r	19	if-equ-X
6	sg-move	20	sg-rotate-r
7	if-less	21	h-copy
8	h-search	22	sg-sense
9	sg-sense	23	nop-C
10	if-equ-X	24	if-label
11	pop	25	nop-C
12	h-copy	26	nop-B
13	sg-rotate-l	27	if-label
14	if-grt-X	28	mov-head

Figure 34 shows the AMTQ distribution from the GT, RL-GT, CF-GT, and the one that was seeded from the Right and Left Turns experiment, the one that was seeded from the Cue Once experiment, and the GT from RL waterfall experiment. The performance of the different

experiments is statistically different. (Kruskal-Wallis $p=2.14 \times 10^{-7}$). Figure 35 shows that the control treatment (GT) is significantly different to the CF-GT and the CT form RL waterfall using the Bonferroni adjustment to do multiple comparisons,

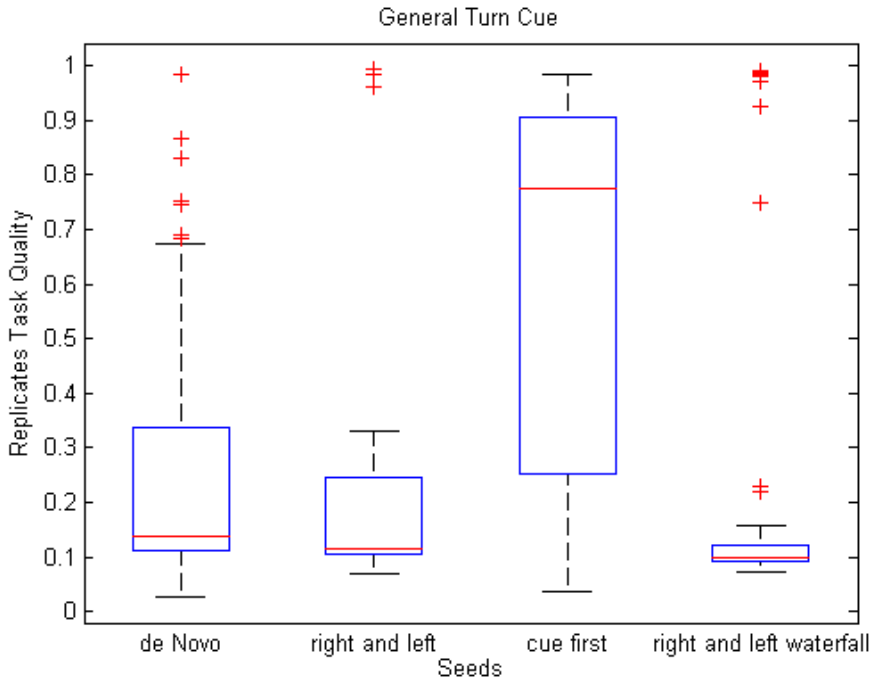


Figure 34. The TQ distribution from the GT, RL-GT, CF-GT, and the GT from RL waterfall experiment

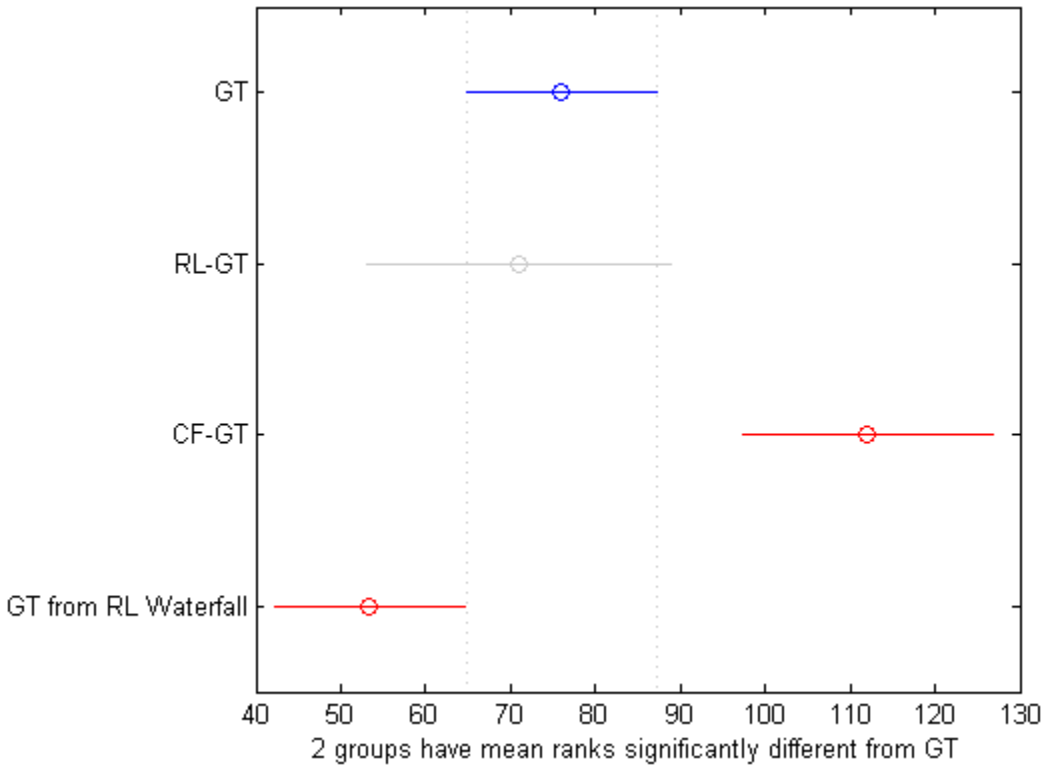


Figure 35. Bonferroni adjustment for multiple comparisons for the General Turn experiments. The Control treatment (GT) is significantly different to the CF-GT and the CT form RL waterfall.

GT from CF Waterfall Transplants

The general turn experiment form the cue first waterfall transplant (GT from CF Waterfall) the last of the sets. The organisms used to seed this last experiment come from the top five organisms that evolved in the CF, which in turn uses as seeds the top five organisms from the RL. This RL uses the top five organisms that evolved from the ST. Figure 36 shows the AMTQ of all the GT experimental sets. The GT from CF waterfall organisms' AMTQ starts increasing quickly as evolution progresses; by update 2,000, the organisms reach an AMTQ of 0.4, and the AMTQ slowly increases to 0.5 at the end of the experiment. This experiment has the second highest AMTQ of all of the GT experiments. Table 22 shows the top five performing replicates of the GT from CF waterfall experiment.

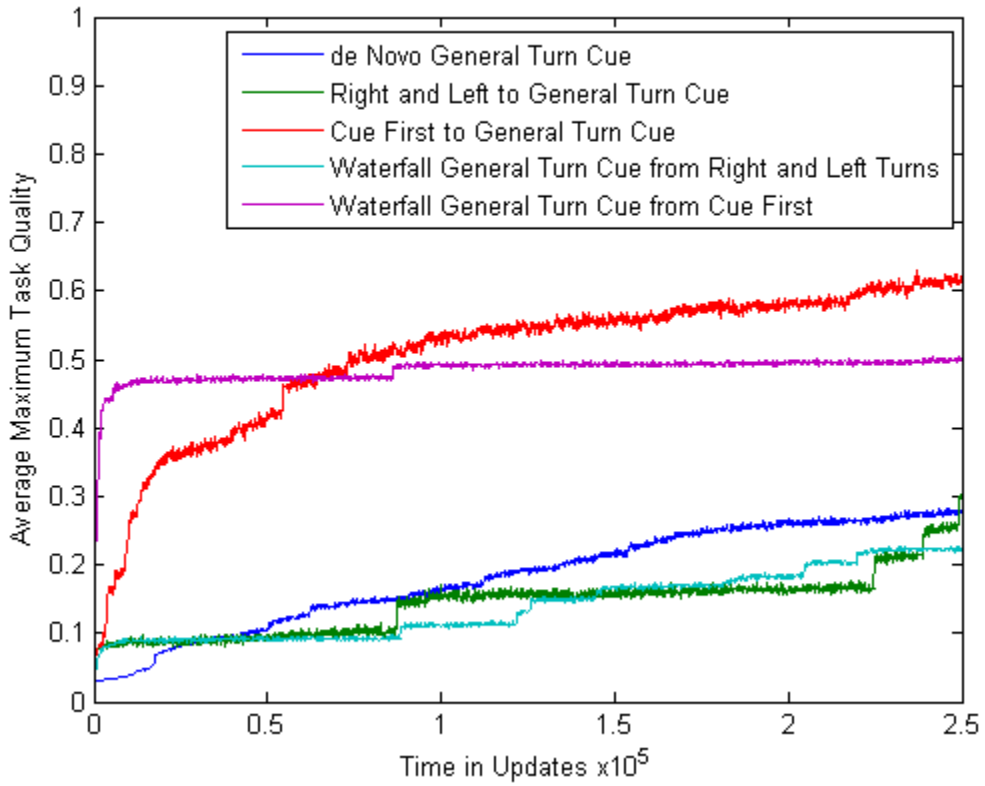


Figure 36. The average maximum task quality (AMTQ) for the GT experiments. The GT from CF waterfall performs better only at the beginning, but then plateaus.

Table 23. The top five performing replicates from the Waterfall General Turn Cue experiments seeded from the Cue First experiment.

Replicate	TQ	Seed
3	0.992695	1
26	0.962294	3
16	0.961403	2
13	0.960466	2
18	0.95961	2

The following is the pseudocode of the most fit organism in the waterfall experiment using a CF first seed organism.

```
do
     IF(BX > 0) swap BX <-> CX
     rotate right
     move
     sense
     if(BX == 1) BX = BX + CX
     copy
     rotate left
     If (BX > 2) rotate left
     BX--;
     If (BX == 1) rotate right
     Copy
     Copy
While (BX != -1)
```

The first instruction is a comparison to see if the value stored in the BX register is greater than 0. If this condition is true, the BX and CX registers values are. This swap instruction maintains control of the direction the organisms turn when it encounters a general turn cue. Then the organism rotates right, moves, and performs a sense. The values of the BX and CX registers get added, and the result is stored in the BX register when the organism senses a general turn cue (value of 1).Then the organism copies one instruction for the child organism and rotates left. Another comparison is made to see if the value is greater than a right turn cue (value of 2), and the organism performs a left turn if this test is true. Then the value in the BX register is

decremented by one, and it is compared to the value 1 to determine if the organism has encountered a right turn cue and needs to make a right turn. Finally, the organism copies two instructions before going back to the beginning of the module. The loop repeats until the organism senses an empty cue. The decrement instruction will make the BX register value -2, causing the main module to exit. The organism then will continue replicating, and divide later in the genome. Table 24 shows the Avida instructions for the main module of the fittest organism in the GT from CF waterfall experiment. The organism's main module is composed of 22 instructions. The total genome is made of 355 instructions. Figure 34 shows the TQ distribution from the GT, RL-GT, CF-GT, GT from RL waterfall, and GT from CF waterfall. The performance of the different experiments is statistically different. (Kruskal-Wallis $p=8.63x10^{-7}$). Figure 38 shows that the control treatment (GT) is only significantly different to the CF-GT once we include the CT form CF waterfall using the Bonferroni adjustment to do multiple comparisons.

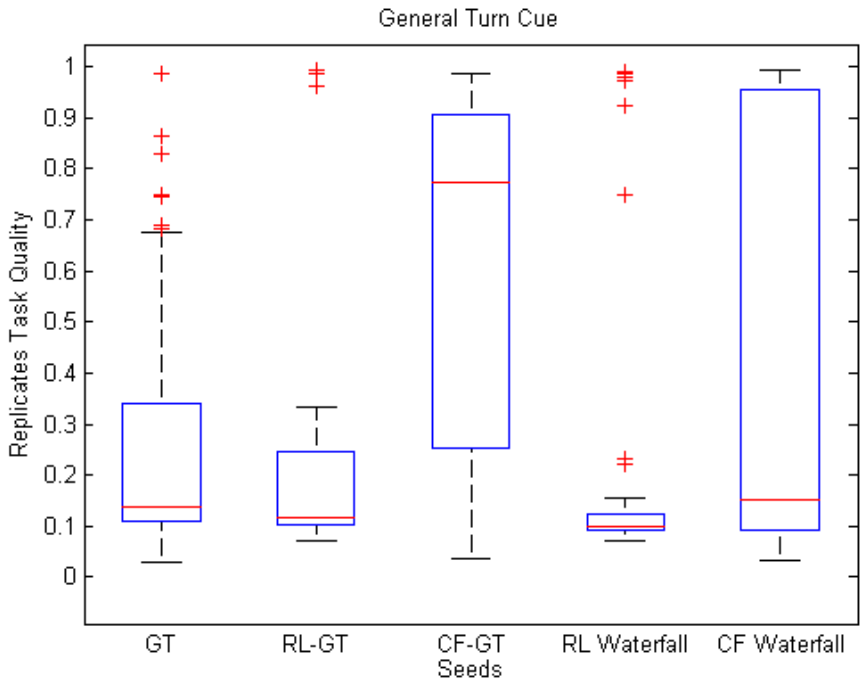


Figure 37. TQ distribution from the GT, RL-GT, CF-GT, GT from RL waterfall, and GT from CF waterfall.

Table 24. Avida instructions for the main module of the fittest organism in the GT from CF waterfall experiment.

Main Module			
Position	Instruction	Position	Instruction
252	push	263	if-grt-X
253	if-grt-0	264	nop-B
254	swap	265	sg-rotate-l
255	push	266	dec
256	sg-rotate-r	267	if-equ-X
257	sg-move	268	sg-rotate-r
258	sg-sense	269	h-copy
259	if-equ-X	270	h-copy
260	add	271	if-n-equ
261	h-copy	272	nop-A
262	sg-rotate-l	273	mov-head

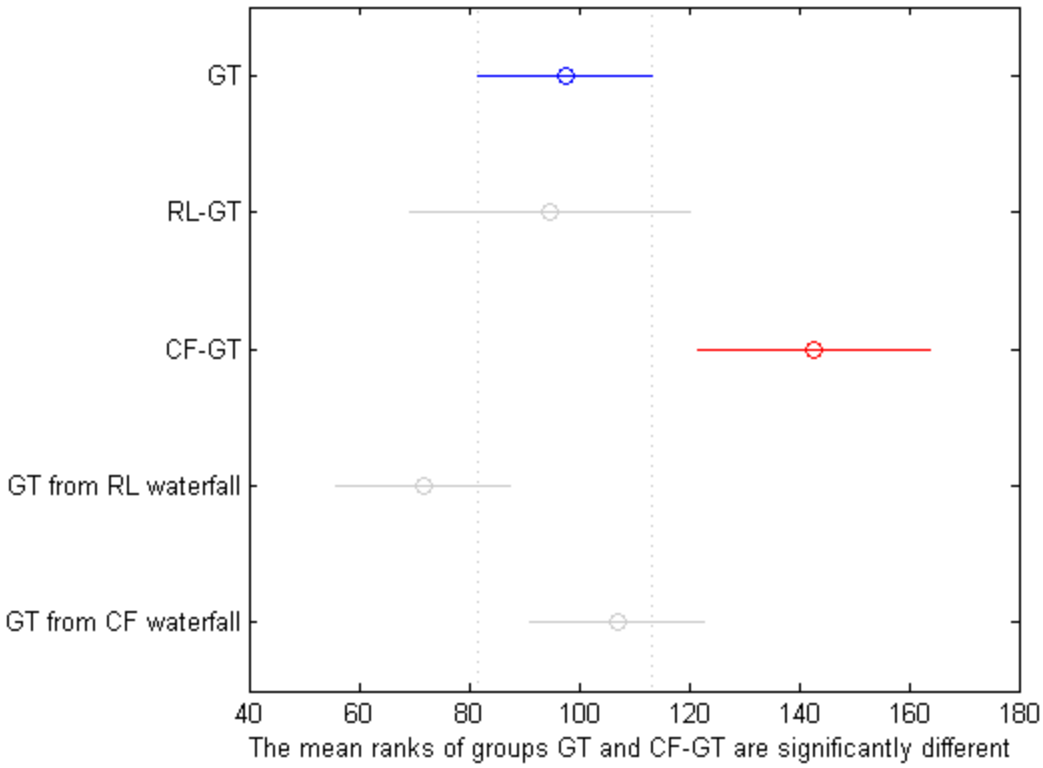


Figure 38. Bonferroni adjustment for multiple comparisons. Control treatment (GT) is significantly different only to the CF-GT once the CT form CF waterfall is introduced.

Results

Table 25 shows a summary of the Waterfall Transplant experiments, the rank of the organism where they came from, and the number of replicates that had a task quality (TQ) above 0.9. These were the first set of experiments where all the transplanted organisms survived. The top performing organism from GT from RL waterfall experiment is made of 476 instructions. It has doubled the number of instructions compared to RL-GT. The main module also increased in the number of instructions from 24 to 28. The genome for the top performing organism from the GT from CF waterfall experiment is made of 355 instructions, and the main module is comprised of 22 instructions. It seems like the organism did not increase the size of the main module is because evolution modifies adaptations previously encountered instead of generating new modules that have the adaptations to deal with the environments. This gives the organisms an advantage over other organism that have adaptations the might create a new section for the organism to deal with the environment.

Similar to the One Level transplant organisms, the Waterfall environment that included the CF organism had better performance, and a greater amount of organisms that had a TQ above 0.9. In both waterfall treatments the size of the genome increased, also the number or replicates that had a TQ above 0.9 increased.

Experiment	Top Replicate	Top 5 seed	Replicates with TQ > 0.9
GT from RL Waterfall	27	3	6
GT from CF Waterfall	3	1	23

Table 25. The Waterfall transplant organism experiment with the top performing replicate, the rank of the organism it came from, and the number of replicates that had a task quality (TQ) greater than 0.9.

CHAPTER III.

CONCLUSION AND FUTURE WORK

Conclusion

In this thesis I focused on investigating how organisms might evolve complex behaviors, and how the complex behaviors affect the genome complexity. Using the Avida state grids, I demonstrated that although an organism can evolve adaptations from scratch, it is more difficult to do so. First I implemented a set of baseline experiments that provided how well the organisms adapted to the different set of environment configurations. It was unexpected to see that organisms that encountered only one directional cue in their lifetime evolved to deal with environments that had both directional since they never encountered environments with right and left cues instead of separate modules where each module specializes in a single type of turn like the CF treatment. The most fit organism from each environment also varied a lot in size (from 160 to 399), but the main modules that handled the path traversal task for their respective environment had similar sizes.

In the second set of experiments I transplanted the top five organisms from one environment to the next most complex one. This showed how the organisms might evolve new more complex adaptations. In none of the experiments all of the transplanted organisms survived. The experiment in which the most transplanted organisms survived was from the ST-RL experiment with four out of five organisms finishing the experiment. The ones with the least number of surviving descendants came from RL, where only two of the five transplanted

organisms were able to replicate. The analysis from these experiments showed that organisms modified previous adaptations instead of creating new ones. They also demonstrated that in most cases, the fittest organism from one environment is not capable to create new adaptations to produce the fittest organism in another environment. Another finding was that organisms that encounter the same cues in a previous environment, with different arrangements, were better equipped to survive in the new environment. The last thing I observed here is that the better adapted an organism is to its environment the greater the instruction length. This is true even if the main module did not increase the number of instructions significantly.

In the last set of experiments I transplanted the top five organisms from the simplest environment to the most complex one. Here the results showed how organisms that had more time to evolve are able to survive in the new environments. Even if the AMTQ kept diminishing, the amount of organisms that survive the transplant to the new environment was greater. It also confirmed the result from the previous set of experiments: if an organism encounters the same cues in a new environment in a different arrangement, it evolves an adaptation faster, and more descendants have a better task quality.

Future Work

The focus of this project was to get some insight into how complexity arises in evolution. I observed how adaptations affect the overall size of the organisms and the main module size. I also observed that the organisms encapsulated the instructions used for the path traversal task into functions, and how these functions affect future adaptations.

Historical Contingency

Historical contingency means investigating how accidental changes in the genetics of a population affect the path of evolution. It would be interesting to see how historical contingency

affects the evolution fitness of the different organisms. Usually the fittest organism from an environment did not come from the fittest in another environment. If we take the transplant experiments as an example, how much would a task quality of an organisms increase by the end of an experiment if the environment changes to a more complex one sooner before the organism is not able to adapt to a new environment. The frequency that an environment can change would help identify a threshold where the most organisms are capable of surviving an environmental change.

Genome Length Limits

Another thing that can be researched further is what limitations the length of the genome can have in evolution. If the cost of the instructions increases, or if the task merit decreases, the task quality of a solution may be affected. It would also be possible to see how the genome length affects the appearance of new adaptations.

Instruction Encapsulation

All of the tasks in this thesis are related, but what if the environment is changed to perform different functions at different times, or if we reward two or more independent tasks? How would such a situation affect future adaptations? Would it be possible to develop two types of competing adaptations? Or would evolution produce an organism that can perform both functions depending on the situation?

This work is only a small portion of the research that is being done to understand how complexity can appear in different systems. Here I have performed experiments on an evolutionary system to comprehend some of the biological principles that could be applied to different areas of study. I also mentioned some of the possible directions for future investigation that could expand the work presented here.

REFERENCES

Back, T., Hammel, U., & Schwefel, H. P. (1997). Evolutionary computation: Comments on the history and current state. *Evolutionary computation, IEEE Transactions on*, *1*(1), 3-17.

Brelles-Marino, G., & Bedmar, E. J. (2001). Detection, purification and characterisation of quorum-sensing signal molecules in plant-associated bacteria. *Journal of biotechnology*, *91*(2-3), 197-209.

Cordon, O., Gomide, F., Herrera, F., Hoffmann, F., & Magdalena, L. (2004). Ten years of genetic fuzzy systems: current framework and new trends. Fuzzy sets and systems, 141(1), 5-31.

Darwin, C. (1859). On the origins of species by means of natural selection.*London: Murray*.

Dennett, D. C. (2002). The new replicators. *The Encyclopedia of Evolution*, *1*, E83-E92.

Friedberg, R. M. (1958). A learning machine: Part I. *IBM Journal of Research and Development*, *2*(1), 2-13.

Grabowski, L. The Evolutionary Origins of Memory Use in Navigation. 2009.

Grabowski, L. M., Bryson, D. M., Dyer, F. C., Ofria, C., & Pennock, R. T. (2010). Early evolution of memory usage in digital organisms. In *Artificial Life XII: Proceedings of the Twelfth International Conference on the Synthesis and Simulation of Living Systems* (pp. 224-231).

Holland, J. H. (1962). Outline for a logical theory of adaptive systems. *Journal of the ACM (JACM)*, *9*(3), 297-314.

Koch, C., & Laurent, G. (1999). Complexity and the nervous system. *Science*,*284*(5411), 96-98.

Lenski, R. E., Ofria, C., Pennock, R. T., & Adami, C. (2003). The evolutionary origin of complex features. *Nature*, *423*(6936), 139-144.

Ofria, C., Adami, C., & Collier, T. C. (2002). Design of evolvable computer languages. *Evolutionary Computation, IEEE Transactions on*, *6*(4), 420-424.

Ofria, C., Adami, C., Collier, T. C., & Hsu, G. K. (1999). Evolution of differentiated expression patterns in digital organisms. In *Advances in Artificial Life* (pp. 129-138). Springer Berlin Heidelberg.

Zhang, S. W., Bartsch, K., & Srinivasan, M. V. (1996). Maze learning by honeybees. *Neurobiology of learning and memory*, *66*(3), 267-282.

APPENDIX A

APPENDIX A

Table 26. Instruction set used for experiments. Instructions in italics are the instructions that allow organisms to move.

Instruction	Description
nop-A	This instruction does not perform an operation by itself, but it can be used to modify the operation of an instruction.
nop-B	see description of nop-A
nop-C	see description of nop-A
If-n-equ	Compares the contents of the BX register to its complement. If the two values are equal, the instruction after the nop label is executed. Otherwise it is skipped.
If-less	Compares the contents of the BX register to its complement. If the value in the BX register is less than the compared value, the instruction after the nop label is executed. Otherwise it is skipped.
Pop	Remove the top item from the active stack and save it in the BX register
Push	Copy the current value of the BX register and place it as a new entry at the top of the active stack
Swap-stk	Change the active stack
Swap	Exchange the value in the BX register with the value of its complement
Shift-r	Shift the bits of the value in the BX register to the right
Shift-l	Shift the bits of the value in the BX register to the left
Inc	Increment the value of BX by one
Dec	Decrement the value of BX by one
Add	Add the values of the BX and CX registers and store the result in the BX register
Sub	Subtract the values of the BX and CX registers and store the result in the BX register
Nand	Perform a bitwise NAND operation using the values in BX and CX registers and save the result in the BX register
IO	Output the value of BX, check for any tasks performed, and input the new value into BX
h-alloc	Allocate new memory for the organisms, up to the maximum it is allowed to use for its offspring
h-divide	Divides off an offspring. The parent organism keeps the state of its memory to the read-head. The offspring's memory is initialized to the contents of memory between the read-head and the write-head. Any memory past the write-head is removed.

Continued on next page

Instruction	Description
h-copy	Copies the contents of the organism's memory at the position of the read-head to the position of the write-head. If the copy mutation rate is non-zero, a random instruction will be placed at the write-head according to the mutation probability.
h-search	Reads the label that follows the instruction and finds the location of a complement label in the code. BX is set to the distance from the current IP position to the complement, and CX is set to the size of the label. The flow-head is placed at the beginning of the complement label. If no label follows, set BX and CX to zero, and place the flow-head on the instruction immediately following the h-search.
Mov-head	Moves the IP to the position of the flow head
Jump-head	Reads the value in the CX register and moves the IP that fixed amount in the organism's memory
Get-head	Copy the current position of the IP to the CX register
If-label	Read in the label following the instruction If the complement of the label was the most recently copied series of instructions, execute the next instruction, otherwise skip the next instruction.
Set-flow	Move the flow head to the position in memory specified by the value stored in CX
If-grt-0	Compares the contents of the BX register to 0. If the value is greater than 0, the instruction after the nop label is executed. Otherwise it is skipped.
Sg-move	Moves to the cell the organisms is currently facing.
Sg-rotate-l	Rotates the facing of the organism 45 degrees counter-clockwise
Sg-rotate-r	Rotates the facing of the organism 45 degrees clockwise
Sg-sense	Returns the value of the state in the current cell
If-grt-X	Compares the contents of the BX register to a fixed value determined by the modifying NOP label (No label = 1, nop-A = -1, nop-B = 2, nop-C = 4). If the value in BX is greater than the value compared to, the instruction after the nop label is executed. Otherwise it is skipped.
If-equ-X	Compares the contents of the BX register to a fixed value determined by the modifying NOP label (No label = 1, nop-A = -1, nop-B = 2, nop-C = 4). If the value in BX is equal to the value compared to, the instruction after the nop label is executed. Otherwise it is skipped.

BIOGRAPHICAL SKETCH

Javier Antonio Magaña was born September 1983 in Mexico City. He moved to Chihuahua, Mexico at age 5. He attended the Dozal Bilingual Elementary School (Escuela Bilingüe Dozal) from September 1989 to May 1995 and Technological Institute of Superior Studies of Monterrey in Chihuahua (Instituto Technológico de Estudios Superiores de Monterrey, Campus Chihuahua) from September 1995 to May 2001. In 2001 he attended the University of Texas-Pan American (UTPA). In 2006 he received his Bachelor of Science in Computer Science and started to work as a programmer analyst in UTPA until 2012. By May 2012, he was hired by Wal-Mart in Bentonville Arkansas as a Programmer Analyst. He obtained his Masters of Science in Computer Since in May 2013. For questions or comments about this work, he can be reached at javieramd@gmail.com