This is to certify that the
thesis entitled

EVOLUTIONARY DYNAMICS OF 3D DIGITAL CONSTRUCTS

presented by

JASON MICHAEL STREDWICK

has been accepted towards fulfillment
of the requirements for the

Master of Science	degree in	Computer Science and Engineering

Major Professor's Signature

12-6-05

Date

EVOLUTIONARY DYNAMICS OF 3D DIGITAL CONSTRUCTS

By

Jason Michael Stredwick

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Computer Science and Engineering

2005

ABSTRACT

EVOLUTIONARY DYNAMICS OF 3D DIGITAL CONSTRUCTS

By

Jason Michael Stredwick

Convergence and stagnation are two of the biggest obstacles to the development of open-ended, evolving systems. These issues have received much attention in both traditional biological research and in the field of Evolutionary Computation. Here I study their effects on evolution in 3D virtual worlds, with a focus on the evolving population's ability to continuously produce new, nontrivial survival strategies. I pay particular attention to measures of convergence, stagnation, and fitness, which provide an avenue for comparison among distinct evolving systems. Toward this goal, I designed and utilized an evolutionary system called WhirlingDervish that evolves 3D morphology and behavior to examine these issues.

Convergence and stagnation are clearly related qualities. I define convergence as the reduction in genetic diversity among individuals in a population, and stagnation as the population's lack of progress towards the evolutionary goal over a period of time. The process of measuring these qualities in a population draws upon methodologies commonly used in biology and evolutionary computation, including measures of genetic diversity and both genetic and phenotypic variation. I test how informative these methodologies are toward approximating the level of continued evolution in WhirlingDervish and make use of their results to compare potential fitness functions. Finally, I look for correlations between these measurements and the structural and behavioral qualities of individuals in the final population.

ACKNOWLEDGEMENTS

I would like to thank Dr. Charles Ofria for taking me on as his student, introducing me to the fascinating area of computational evolution, and suggesting the idea for the WhirlingDervish system. I also appreciate all his time and guidance throughout my learning experience. Most of all, I really appreciate his patience.

I would also like to thank the other members of my committee, Bill Punch and Eric Torng, for their time and effort, and their patience with me as I attempted to finally set a date for my defense.

Thanks to Matthew Hall for all his support and insight.

Many thanks to all of the people in both the Ofria and Lenski labs for all of their help and support. To Dehua Hang for all of the problem discussions, and helping me make the most out of my time (I had a lot of fun). Wei Huang gave me a lot of support during my time as a Masters student along with useful feedback on this thesis and many other problems I tackled. I would like to thank Kaben Nanlohy for introducing me to QT and all of the discussions about interesting and complex data structures and coding practices. I also appreciated the feedback on the WhirlingDervish system from Art Covert, along with his feedback on my thesis. Finally, I would like to thank Brian Baer for all of his efforts to keep our systems running smoothly and his extensive system administration knowledge.

The CSE department and the National Science Foundation (Grant DEB-9981397 to Lenski, Adami, and Riley) for each provided me with three semesters worth of funding, which made it possible to complete my Masters degree.

Finally, I would like to thank my wife, Kristina Stredwick, for all of her support and encouragement. I would also like to thank her for all of the time she spent discussing the many nuances of evolutionary theory with me and helping me bridge some of the gaps between biological and computational evolution.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

"Images in this thesis/dissertation are presented in color."

Chapter 1 Introduction

In this thesis I utilize an evolutionary system called WhirlingDervish to examine the use of diversity and variation measures to characterize convergence and stagnation in systems that evolve 3D morphology and behavior. I then apply the measures to each set of fifty runs for six variants of WhirlingDervish's default fitness function in order to study the effects of the different phenotypic traits that comprise the default fitness function. The characterization for each set of runs provides me with a means to compare the variants and differentiate the evolutionary impact of the various fitness factors used within the default fitness function. Then I will examine how closely the diversity and variation measures approximate the evolved behaviors for six case studies. My goal for these analyses is to provide a means of comparing results within studies in the same system as well as providing a common set of measurements for comparisons between related systems.

Before analyzing the diversity and variation measures, it is important to understand the system that I use for a testbed. In Chapter 2, I provide a detailed overview of the WhirlingDervish system, and compare it to other related programs. In WhirlingDervish, 3D structures are evolved that are capable of moving about their virtual world. Individuals in this system do not have a brain (or other centralized control structure) to regulate their movements. The system employs simplified physics and phenotype models while maintaining a 3D virtual world that is intuitive to the user. The simplifications allow for large population sizes capable of running in real-time on a standard desktop machine. Despite the simplifications in the system, it still has the potential to produce highly complex results. The dynamic genetic language provides an

infinite number of potential solutions, although the types of solutions may be categorized into a finite number of groups.

The following are some highlights that define WhirlingDervish as an evolutionary system. It uses a simple, assembly-like language for its genetic representation, which is executed by an external structure in order to generate the phenotype (an individual's body and, consequently, its behavior). When replication occurs, the offspring fitness is computed using an explicit function. Fitness parameters are measured after testing the individual in its virtual physical world. The goal for the system is to evolve individuals that are capable of consistent movement, while avoiding premature convergence to undesirable solutions such as "one-shot" constructs or other evolutionary dead-ends. One-shot constructs have an initial burst of movement and then never move again. Typical examples are individuals that move by simply falling over or by throwing themselves a certain distance and collapsing into a heap when they land.

In Chapter 3, I use the WhirlingDervish system to generate fifty replicate runs using the default setup. I measure the diversity and inter-individual variation within each population and use the results to characterize convergence and stagnation. By examining the supporting information such as the relative proportion of neutral, beneficial, and detrimental mutations, I explore the evolutionary dynamics and judge the usefulness of these measures for assessing convergence and stagnation.

Since the goal of this study is to assess the usefulness of various diversity and inter-individual variation measures, I sample a broad range of measures. The majority of the measures are drawn from genetic programming while some come from biology and genetic algorithms. Richness, Shannon Entropy, Simpson's Diversity Index, and

McIntosh's Diversity Index are the four measures of diversity that I use and apply at both the genotype and phenotype levels. These measures rely upon the classification of individuals into groups and the proportion of the total population represented by each group. Conversely, variation measures utilize statistics between individuals rather than groups. I use four categories of variation: genetic variation, phenotypic variation, average inferiority, and measures relating to the most recent common ancestor (MRCA). Genetic variation quantifies the difference between two genomes across a population over time. Phenotypic variation examines the change in phenotypic traits, including fitness, across a population over time. Average inferiority is similar to phenotypic variation for fitness, but measures the average difference from the fitness of the most-fit individual rather than a straight average of fitness. The measures that relate to the MRCA include the generation that the MRCA was generated and the average genetic difference between the MRCA and each individual in the final population.

In Chapter 4, I utilize diversity and variation measures to discover the impact of each component in WhirlingDervish's default fitness function, which I call the Base fitness function. The experiment conducted here examines the impact of these factors by knocking out each fitness component, one at a time, and generating fifty replicate runs. I also generate fifty replicate runs for a fitness function that only utilizes the single factor of distance moved, as this function is used by other related evolutionary systems. By comparing the convergence and stagnation for each set of runs and their ability to move, I can approximate the impact of each component to the Base fitness function and isolate its contribution to the evolutionary process.

In Chapter 5, I seek to understand how informative the diversity and variation

measures are for approximating structural and behavioral variation. It is exceptionally difficult to automatically perform qualitative measures on morphology and behavior, but on a limited scale this can be done using visual analyses. To gain a better insight into how the diversity and variation measures at the genotypic or phenotypic level relate to *behavioral* variation, I perform six case studies by choosing six runs from the fifty used in Chapter 3. For each case study, I first describe the most-fit individual. Next, I visually examine samples from the final population, the MRCA for the final population, and a select number of individuals along the lineage prior to the MRCA, and determine what, if any, variance exists. This information approximates the variation in the final population as well as the variation that has occurred over time. Finally, in this chapter I examine an alternative genetic variation measure that examines the variation between two genomes after all neutral genomic positions have been removed. This measure was not been included in Chapter 3 due to time constraints, but the information found here provides an idea of the impact of neutral mutations on measuring genetic diversity and variation.

Chapter 2 WhirlingDervish, A System for Evolving 3D Morphology and Behavior

I designed and implemented the WhirlingDervish software to be a flexible research platform for evolutionary studies, which currently targets the evolution of 3D morphology and behavior. The purpose of generating a new system rather than using a preexisting one is the ability to have large population sizes that run in real-time on a standard desktop machine. WhirlingDervish accomplishes this feat by using computationally simple physics and reducing the complexity of the phenotype model to a wire-frame body composed of point-masses and springs. Additionally, individuals lack a behavioral control structure such as a neural network, which is often a central facet of related systems, but creates unnecessary complexity for simpler studies.

The inspiration for the wire-frame, mobile body comes from The Soda Play software (Soda Play, www.sodaplay.com). Soda Play is an online application that allows users to create a two-dimensional, wire-frame body that moves around in a simple virtual environment. The website contains many pre-made individuals that exhibit a wide range of morphologies and behaviors such as walking and rolling. WhirlingDervish expands these structures into three-dimensions, and successfully generates mobile individuals through an evolutionary process.

The WhirlingDervish system uses a simple evolutionary algorithm to evolve individuals capable of motility. Each generation begins by simulating all of the individuals in a virtual environment and measuring characteristics of each individual (as described in Section 2.3.1). These measurements are used to compute the fitness value. Next, the individuals are selected for replication using fitness proportional selection. As

each individual is selected, replication occurs with a chance of mutation but no crossover. Once an entirely new population is generated, the process starts over again until the last generation is complete.

While the evolutionary algorithm is fairly standard, it employs a complex evaluation method: the simulation of a virtual world, consisting of a single infinite plane (the ground) and the individuals being evaluated. The simulation models the behavior of individuals by managing the point-masses and springs within an individual and the interactions between point-masses and the environment. The simulation of each individual lasts for two minutes at which point, fitness is measured based on data recorded during the simulation.

The rest of this chapter details the WhirlingDervish system and presents its relationship to other systems that evolve 3D morphology and behavior. It begins in Section 2.1 with a description of the evolutionary systems that relate to WhirlingDervish. Then, Sections 2.2 and 2.3 describe the details of the two major subsystems: the virtual world and the evolution structures and processes. Finally, a brief overview of the system's runtime abilities is given in Section 2.4.

2.1 Background

There are many 3D morphology and behavior evolutionary systems currently in use. The evolutionary goals of these systems are to evolve non-trivial behavior and complex morphology. Many of these systems are used to study the evolution of neural networks used as control structures for an individual, bridging the gap between the virtual world and the real world, or to study genetic representations. While each system was created by different groups of researchers, they share many aspects in common.

One commonality is the phenotype structure that consists of rigid components connected by joints. These are commonly represented as three-dimensional geometric primitives such as spheres, cylinders, or blocks, though the joints are not always visible. Both component types are associated with a number of manipulation functions that cause them to act in a specific manner. A neural network, often called a brain, controls the manipulation of each component, which gives rise to behavior by adding a dynamic quality to the components. In most systems, the structure and the controller are co-evolved.

The inspiration for these evolutionary systems arose from a system called "Virtual Creatures", created in 1994 by Karl Sims (1994a). This system consists of blocks connected through invisible motorized joints, all of which are controlled by a neural network. Sims had a two-fold focus: to explore the possibility of evolving 3D virtual creatures capable of different types of movement and to study methods of evolution under direct, one-on-one competition. In Sims's initial work, he was successful at using a standard evolutionary algorithm with a static fitness function to evolve creatures with fluid, realistic-looking movements. However, subsequent work (Sims, 1994b) suggested that relative fitness through competition rather than treating the evolution of behavior as an optimization problem would create more complex creatures. While he did evolve interesting creatures through competition, he never provided a direct comparison between his fitness schemes, most likely due to the differences in evolutionary goals.

The system created by Tim Taylor and Colm Massey (2001) was a recreation of Sims's system. Its purpose was to test the use of off-the-shelf physics engines, more specifically MathEngine, as an alternative to individually crafted physics engines.

During the creation of their system, they encountered and documented many of the common problems faced by designers of these types of systems, mostly dealing with premature convergence. One such problem is a type of individual they call a "one-push" creature, which is a creature that throws itself a certain distance and then stops moving. They found that "many cases where these 'one-push' creatures were selected in the initial generations, the population reached an evolutionary impasse (a local optimum in the fitness landscape), and had no easy mutational routes to higher fitness" (Taylor and Massey, 2001). They suggest that increasing the time a creature is simulated will reduce the number of "one-push" creatures, but did not indicate if they had tested this suggestion. They also state that Sims was able to achieve continuous swimming over time by giving a stronger relative weight to velocity toward the end of the simulation. However, when they implemented this type of fitness function in their system, they still evolved "one-push" creatures fairly frequently.

Another problem they ran into was "folded" creatures. These creatures have a similar evolutionary effect as the "one-push" creatures where the system would often converge on them if they arose early in the evolutionary process. Folded creatures are described as "creatures evolved that would initially adopt a compact, folded configuration, then as the evaluation period proceeded they would 'unfold' in a particular direction. This unfolding had the effect of shifting the creature's center of mass, thereby increasing its fitness" (Taylor and Massey, 2001). They attempted to prevent this by only "using the distance moved by the body part that had moved least over the duration of the evaluation" (Taylor and Massey, 2001). However, this wasn't enough to completely eliminate the problem.

Based on their evolutionary problems and their attempts to fix them, Taylor and Massey acknowledged an inherent problem with existing fitness schemes. The problem was a trade-off between the inclusion of special cases to prevent undesirable behavior and the ability to use high-level specification of required behavior. They also noticed "even using straightforward fitness functions will *sometimes* produce desired results" (Taylor and Massey, 2001). In terms of preventing convergence, Taylor and Massey suggest that "one way to help prevent the runs getting stuck in local fitness optima would be to maintain the genetic diversity of the population by introducing some kind of incomplete mixing (e.g. by using an island model GA). I have performed some initial experiments with incomplete mixing, but not enough at this stage to say how effective this strategy is at solving the problem" (Taylor and Massey, 2001).

Multiple systems and ideas have arisen from the lab of Jordan Pollack. Some of their goals are truly novel, including evolving individuals capable of being built as functional structures in the real world (Lipson and Pollack, 2000; Hornby et. al. 2001a; Pollack et. al., 2003). Ideas centering on language and language structures have been long standing directions of research within the lab. One of the general language constructs is the concept of a Genetic Library Builder (GLiB) (Angeline and Pollack, 1994). GLiB is a language paradigm that allows for the dynamic creation of new more complex instructions by associating each new instruction with an ordered set of other instructions that are currently in the library. The library begins as a set of atomic instructions and then mutation operators allows the creation, deletion, and breaking up of instructions within the library. As a genome is copied, it has access to the current state of the instruction library.

The study of language in the evolutionary systems designed in Pollack's lab extended into their systems evolving 3D morphology and behavior (Hornby and Pollack, 2002; Pollack et. al., 2003). They began to use a type of language called Lindenmayer systems (L-systems) as their genetic representation. L-systems use a recursive symbol- and rule-based representation. They were originally created to generate graphics of the branching patterns in plants. By defining a hierarchical set of rules, each assigned a unique symbol; you can build up complexity by composing more complex rules based on other, simpler rules. The use of an L-systems genetic representation in their evolutionary systems allowed for the evolution of symmetrical constructs. They conducted comparisons between the L-system representation and a non-symmetric representation and found that the use of L-systems allowed the creation of more complex and successful constructs.

Another major system that evolves 3D morphology and behavior is called Framsticks (Komosinski, 2003), which was created by Maciej Komosinski and Szymon Ulatowski (1999). The system uses constructs composed of motorized joints and sticks that move using a neural network controller; both the body and the controller are coevolved. Like many of the other systems described here, much of the work on Framsticks deals with the evolution of neural network control structures. The portion of their work relating to the evolution of 3D morphologies and behavior centers around the study of genetic representations, all of which are instruction-based languages that they refer to as 'encodings'. The purpose of this research is to find an encoding that will maximize the complexity of their evolved constructs.

Another system called Artificial Ontogeny (Bongard and Pfeifer, 2003) evolves 3D morphology using spheres as its building blocks, which are controlled by a neural network. The evolutionary goal of this system was to evolve constructs capable of pushing a block, much larger than itself, as far as it could. The most important difference between this system and the other systems mentioned above is that it does not use a genetic language to build the phenotype. Instead it uses an alternative genetic representation and mapping algorithm called a gene regulatory network (GRN).

There are other major systems that are similar to those listed above but focus on topics other than strictly morphology or behavior. One example is a system by Bongard and Lipson (2004) that focuses on the co-evolution of simulator and construct for the purpose of automatic design through evolution of robots capable of compensating for bad components or coping with small differences between the real world and the virtual world in which it evolved. There are still other systems that use more rigid phenotypic representations, with specific evolutionary goals. An example is the massive quantity of research being conducted on evolving behavior in static two and four legged constructs, where these constructs are generated prior to evolution, such as the system by Gene Ruebsamen (2002). For more information on 3D morphology and behavior evolutionary systems, some of which were mentioned here, see the paper by Taylor and Massey (2001) and the webpage by Tim Taylor (Taylor, www.tim-taylor.com/research/embody.html).

2.2 The Virtual World

In WhirlingDervish, the virtual world is the subsystem that takes a population of individuals and simulates their actions in a virtual physical environment. The cornerstone of this subsystem is the body of an individual. The body is a three-

dimensional wire-frame structure, composed of point-masses connected by springs that can form complex structures with intricate behavior. This behavior is derived through the interaction of point-masses with the environment and with other point-masses through the springs that connect them. The point-mass is the only physical unit that is capable of environmental interaction; springs are treated as nothing more than forces between point-masses. Point-masses are treated as infinitely small, and therefore cannot collide with each other.

Springs connect pairs of point-masses and apply a force that pushes or pulls them to a specific distance apart (known as the "rest length" of the spring). This force is the only manifestation of the spring. It has no mass or volume and exerts force only along an unbendable straight line between its two point-masses. The rest length of a spring can oscillate over time, adding dynamics to the body, which allows it to move. These oscillations are the primary driving force for the movement, and are effectively a continuous input of energy into the system.

The simulator is the hub of this subsystem. It manages all interactions within the world, particularly the modeling of body dynamics and body/environment interactions, by updating the internal state of a body through the application of physics functions. These processes occur in time steps, meaning that multiple interactions can occur during the same time step. However, the inability of individuals to collide and the environment consisting solely of the ground plane, nullify this issue. Regardless, the entire set of point-masses and springs are updated simultaneously across all individuals for the given time step. The individuals continue to be updated until the maximum simulation time has been reached. At the termination point, all of the relevant characteristics measured

throughout the simulation are saved for use by the evolutionary algorithm in determining the fitness of each individual.

The physics functions are the rules of the system. Point-masses and springs each have a subset of the physics functions associated with them. The functions operate on discrete time steps, such that the state of each point in the next time step can be calculated purely based on the current state of the system. The functions used in the system calculate the forces on each point-mass based on gravity, the forces exerted by springs, ground friction, and air friction (drag). The simulator will then update the positions of the point-masses based on the current forces acting on them through projectile motion. Finally, the system will also update the rest length of each spring, which may alter the spring forces in the next time step. The idea behind altering the spring's rest length comes from the Soda Play software (Soda Play, www.sodaplay.com) and a description of this procedure is fully presented in Section 2.2.3.

2.2.1 The Environment

In order to give meaning to the individuals' bodies, they must have an environment in which to exist. In WhirlingDervish, the environment is a three dimensional virtual world. For the purposes of this project, there was no need for complex topography or objects. Thus the only physical object within the world, besides the bodies of the wire-frame individuals, is the ground. The ground is a flat non-deformable surface that is essentially the infinite xz-plane. This surface provides environmental feedback to an individual's body through its interactions with the point-masses of that body.

The purpose of the ground is to provide a surface on which the individuals can move. To maintain a stable surface, the ground must prevent all point-masses from penetrating through to the other side. If a mass should penetrate the ground during a given time step, I use a specific manipulation to rectify the situation. The manipulation is based on the nature of the movement equation (covered in detail at the end of Section 2.2.2), where each mass has two positions: "current" and "previous". Both the current and previous position's y-components are inverted when the current position penetrates the ground. This inversion represents the point having bounced upon hitting the surface such that it is above ground again. The previous position is also inverted to maintain velocity.

To represent collisions that are not perfectly elastic with the ground, a loss of energy was added to the design. Since the previous position is used to determine velocity, the easiest way to implement energy loss is to modify the previous position relative to the current position. This modification is in the form of a scaling factor (0.4 by default) that is applied to the distance between the two positions. Equation 2.1 describes how the previous position (p_p) is moved closer to the current position (p_c).

$$p_p = p_c + 0.4\left(p_p - p_c\right) \tag{2.1}$$

Also, during collision between a point-mass and the ground, friction must be considered. To allow point-masses to be of different materials I allow each point-mass to have its own coefficient of friction relative to the ground, which I will discuss in the next section. In the future, a more sophisticated friction system could be incorporated into the simulation in order to provide additional accuracy.

2.2.2 Point-mass Physics

The point-mass is one of the basic components of the system. An individual's body is composed of a set of point-masses connected by springs. By definition, a point-mass has a mass (m) but no volume or surface area, and can only move when force is applied to it. The most consistently applied force is gravity. Gravitational acceleration (g) is a vector standardized to <0.0, -9.8, 0.0> m/s^2 and does not change during the simulation. However, the simulator is designed to allow g to be any vector and is modifiable in a configuration file. The force (F) of gravity is given as

$$F = mg \tag{2.2}$$

The other forces that affect a point-mass are friction related. To allow a point-mass to generate friction without a surface area, coefficients of friction: static (μ_s), kinetic (μ_k), and drag (μ_d) are required. These coefficients were tuned by hand to produce realistic results. Since all point-masses are uniform in the current state of the system, they are all assigned the same mass (m) and friction coefficients. These values are changeable from a configuration file, but not during evolution. The standard values are shown in Table 2.1.

Table 2.1. Mass and frictional coefficients.

m	100g
μ_s	0.75
μ_k	0.5
μ_d	0.01

One frictional force is air resistance, which affects all points in motion by providing a force in the direction opposite to the motion and proportional to the velocity. The following equation is used to determine the force of the air resistance, where v is

velocity, v_n is the direction unit vector of the velocity, and μ_d is the coefficient of friction, drag:

$$F = -1.0\mu_d \|v\|^2 v_n \tag{2.3}$$

The other frictional forces comprise ground friction. These forces affect all point-masses that have contact with the ground during a time step. To ensure this frictional force is applied during the same time step that the ground is penetrated, an initial calculation of a point-mass's next position is taken and its y-component is tested for ground penetration. If penetration will occur, the frictional force is computed and applied prior to the actual movement of the point-mass. More complicated approaches to determining when and how frictional forces are applied are available but were unnecessary for the initial design.

There are two types of ground friction, which apply in different situations. The first type is static friction, which uses the coefficient μ_s. This force is only applied to a point-mass when it has no velocity at the beginning of the time step, comes in contact with the ground, and the following inequality is true given the current force (F_c) on the point-mass.

$$\|F_c\| \leq \mu_s m\|g\| \tag{2.4}$$

This force provides a counteracting force that prevents a stationary object from moving along a surface by providing a force equal and opposite to the forces exerted parallel to the plane. Thus the point-mass can only move in the xz-direction when there is force in that direction whose magnitude causes the above inequality to become false. Given the

direction of the current force whose y-component is set to zero (F_d), the equation for the static frictional force is:

$$F = -1.0\|F_c\|F_d \tag{2.5}$$

The other type of ground friction is kinetic friction, which uses the coefficient μ_k. This force only applies when a point-mass collides with the ground and static friction does not apply. Like static friction, it is a counteracting force that applies in the opposite direction of the motion of the point-mass parallel to the ground, which is represented by the velocity. Given the direction of the velocity whose y-component is set to zero (v_d), the equation for kinetic frictional force is:

$$F = -1.0\mu_d m\|g\|v_d \tag{2.6}$$

However, if the magnitude of the frictional force is greater than the magnitude of current force on the point-mass, the following equation is used

$$F = -1.0\|F_c\|v_d \tag{2.7}$$

Once all of the forces have been applied to each point-mass, the simulator determines the new position for each point-mass by using projectile motion. There are a few versions of the projectile motion equation available, but here I used the Verlet algorithm (Verlet, 1967). It relies on the current and previous positions, and the size of the time step. This removes the need to store velocity explicitly, since it is not needed in the equation. However, the velocity and acceleration can be calculated at any time with the data available. The following is the Verlet algorithm:

$$p = 2p_1 - p_0 + \frac{F(\Delta t)^2}{2m}; p_0 = p_1, p_1 = p \tag{2.8}$$

By using only the two positions, the Verlet algorithm reduces error accumulation that can occur as velocity and position data deviate over time. When deviation occurs, it means that one cannot calculate the other using the standard formulas, and the results will continue to deviate by increasing amounts over time. Comparing a time step algorithm using position and velocity with the continuous time version of projectile motion can show the extent of the error.

2.2.3 Spring Physics

Individual movement is driven by two aspects of a spring that control the force exerted on the point-masses that it connects (p_0 and p_1). The first aspect is the simple spring force that attempts to keep the two point-masses a specific distance apart, known as the rest length (ℓ_r) of the spring. The other is an algorithm for something similar to a muscle that essentially imparts a constant inflow of energy into the system that keeps the springs expanding and contracting.

The simple spring force provides an attractive force between the two point-masses when the spring length between them is greater than the rest length of the spring and exerts a repulsive force when the distance is less than the rest length of the spring. The magnitude of the force applied to each point-mass is given by the following equation where k is the spring constant and x is the difference between the current spring length and the rest length:

$$x = \left\| p_1 - p_0 \right\| - \ell_r \tag{2.9}$$

$$F_{mag} = kx \tag{2.10}$$

A force is then applied to each point-mass in the appropriate direction where p_{01} and p_{10} are unit direction vectors from p_0 to p_1 and from p_1 to p_0. The two forces F_0 and F_1 are computed from the following equations:

$$F_0 = F_{\max} p_{10} \tag{2.11}$$
$$F_1 = F_{\max} p_{01} \tag{2.12}$$

The only special case occurs when the spring length is zero. Because the spring does not physically exist and point-masses cannot collide with each other, it is possible for the two point-masses to pass through each other. If a time step begins where the point-masses connected by a spring are at the same position, this can make it impossible to calculate the direction to apply the force. To solve this, I use the difference between the velocities to determine the direction and calculate the magnitude in the normal manner. If the magnitudes of the velocities are both zero, then zero force is applied by the spring. If another force should separate the two point-masses at a later time, the spring will once again be able to provide a force contribution.

Each spring has two main properties that affect the force it exerts. These properties are the rest length and the spring constant. Both of these values are calculated when a spring is first created between two point-masses. The distance between the two point-masses determines the rest length, and the spring constant is determined through the equations shown in Table 2.2:

Table 2.2. Spring Constants

$k = \dfrac{9.8}{\left(1.0 + 0.001\ell_r\right)}$	$\ell_r > 0$
$k = 0.0$	$\ell_r = 0$

The design decision to relate the rest length and the starting length of the spring was made to prevent the springs from exerting huge amounts of force during the first time step of the simulation. In a previous iteration of the system, the rest length had a default value that could evolve. However the wire-frame individual would evolve long starting lengths relative to their rest lengths, resulting in huge differences between the starting and rest lengths. Those types of individuals dominated the evolution of the system and necessitated the change to the current state.

By using only the point-mass and spring forces described thus far, the system will come to a rest if given enough time. In order to prevent this and to inject some dynamic qualities into the system, I added a fluctuation component to the springs. This component essentially adds energy to the system and provides a constant cyclic dynamic to a spring by changing its rest length over time. The change centers on the original rest length (ℓ_0), by following the harmonic motion of a sine wave.

There are two parameters required to define the range of possible rest lengths: maximal multiplier (m_m) and delta multiplier (d_m). The maximal multiplier has a final range of $[-0.45, 0.45]$ and when applied to the original rest length gives the range of rest lengths as $\left[\ell_0\left(1-m_m\right)\ \ell_0\left(1+m_m\right)\right]$. However, the delta multiplier is an evolvable parameter with a range of $[-1, 1]$ that is used to scale the maximal multiplier. By scaling the maximal multiplier, the wire-frame individual is able to adjust the fluctuation range for any of its springs. Figure 2.1 visualizes spring fluctuation.

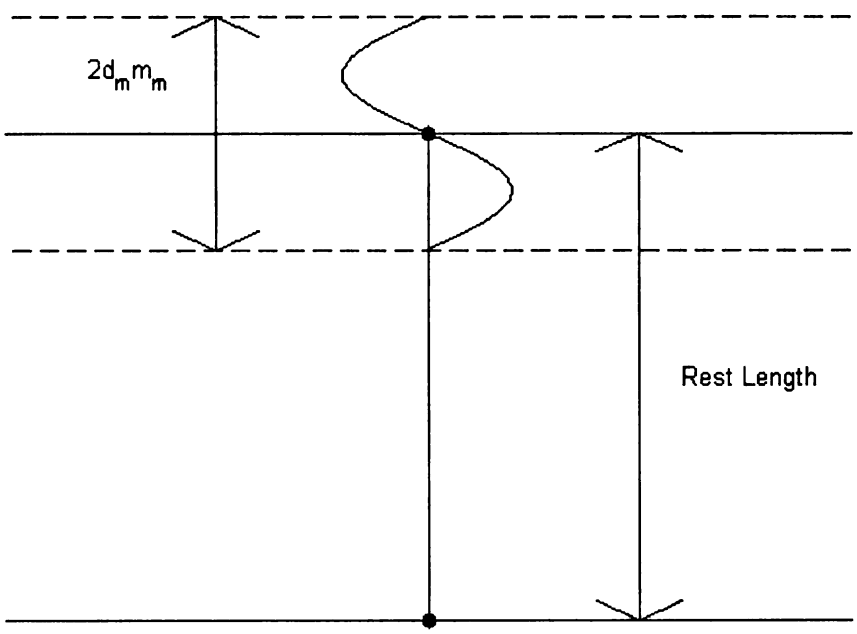


Figure 2.1. An illustration of spring fluctuation. The line between the two red points is the spring at rest length. The dashed lines represent the range of possible lengths through which the spring can move. The fluctuation utilizes a sine wave for the transition rate over time, where the maximum possible length change is twice the maximal multiplier times the delta multiplier.

The fluctuation range is represented as the multiplier (Ψ). The absolute value of the delta multiplier is necessary to ensure the positive value of Ψ. The value of Ψ is determined with the following equation:

$$\Psi = |d_m| m_m \tag{2.13}$$

Once the range of possible rest lengths are defined, a method of traversal is required to give the cyclic change of the rest length. A sine wave was chosen for simplicity and the output range of $[-1, 1]$. Three parameters are required to generate the wave: rate (r), where along the wave a spring starts (s), and time (t). The rate and

starting position each have a range of $[-1, 1]$ and are evolvable, while time can be any value but is scaled to a range of $\left[0, \frac{2\pi}{\|r\|}\right]$. The phase ($\Phi$) is simply a multiplier that when applied to Ψ determines the correct scaling factor for the original rest length. The equation for the phase follows:

$$\Phi = \sin(rt + \pi s) \tag{2.14}$$

The change in rest length is given in the following equation:

$$\ell_r = \ell_0(1 + \Phi\Psi) \tag{2.15}$$

2.3 Evolutionary Structures and Processes

The evolutionary structures can be broken down into two components: the population and the genome. The population is a collection of all of the individuals in a given generation. While the virtual world and all of the evolutionary processes affect each individual, they perform their functions at the population level. The genome is an array of instructions used to create the phenotype. Some instruction examples are creating a new point-mass, changing the properties of a point-mass, connecting two point-masses together with a spring, or performing other sorts of calculations or flow control. Some instructions have parameters, where a parameter is either a vector of three real numbers or a reference to a vector within the construction object (see Appendix B for further information on the construction object). The other important aspects of the evolutionary algorithm are the fitness scheme (Section 2.3.1), the genetic language (Section 2.3.2), and replication and mutation (Section 2.3.3).

2.3.1 Fitness Scheme

The fitness scheme is both the method of determining fitness (ie, the fitness function) and its interpretation. In biology, fitness is measured as the impact that a genotype has on future generations, often in terms of replication rate. Evolutionary computation considers fitness explicitly as the quality of a solution, whether absolute or relative to others in the population. There are many methods available to compute fitness such as an explicit fitness function, the outcome of direct competition, or replication rate. Determining the appropriate fitness scheme for a given problem is important because the scheme can bias the solution, affect evolutionary speed, and change the likelihood of premature convergence. The interpretation of fitness is the method used to determine the number of offspring each individual passes on to the next generation.

My goal in designing the fitness function was to evolve individuals that continuously move over time. However, it is difficult to design a static fitness function that can encompass the abstract nature of the goal in concrete terms. For example, I could use a function to measure fitness as the difference between the center of mass at the beginning and end of the simulation. Unfortunately, this function will encourage the evolution of individuals that move either by flinging themselves and collapsing flat, or else by simply growing tall and falling over. Thus the fitness function must not allow the population to easily get stuck at these less desirable solutions. This trade-off between restricting the fitness function versus using high-level descriptions to define the function had already been found by Taylor and Massey (2001).

The fitness function I use in the WhirlingDervish system returns a single real number value. This value is an accumulation of the raw values from the following

phenotypic trait measurements: xz-distance moved by the center of mass, the height of the center of mass, the overall height of the body, and the total number of point-masses in the structure. All four of these measurements have restrictions. The xz-distance only contributes to the fitness if the final height is at least one third of the starting height. For the two height measurements, neither can contribute more than nine meters, and the number of point-masses is capped at ten. While these restrictions affect fitness, none of them impact structure or behavior directly. Also, while there are restrictions there are also scaling factors. Both the center of mass height and the xz-distance moved are scaled by a factor of three, while the number of point masses are scaled by a factor of 0.0254.

While there are additive elements to the fitness function, there are also a few detrimental elements. The primary detrimental element is the cap on the number of springs. If the number of spring used in the body of an individual exceeds fifty-four, the fitness is reduced by one for each spring over that cap. The number fifty-four is derived as one less than the minimum number of springs to fully connect eleven point-masses. The other caps put restrictions on the structure as a whole, such as a maximal bounding box, in order to prevent outlandish structures and behaviors. Before these restrictions, it was common to evolve individuals that were many thousands of meters tall.

2.3.2 Genetic Language

Genetic languages play an important role in the evolution of morphology because they provide a method of describing how to build a body, which is more expressive than a direct encoding. A direct encoding explicitly lists each body component and all of its parameters. There are many advantages and disadvantages to genetic languages, mostly revolving around what is possible through a single mutation. One advantage is that a

language can be designed to have high-level concepts represented as a single atomic unit, thus a mutation can produce large, meaningful changes. On the other hand, this can also be a disadvantage because the particular choice of instructions invariably also limits the types of changes possible.

The power of a genetic language is that the designer has the ability to implement abstract concepts in a language that would otherwise be difficult to evolve through natural selection. A simple example is that of a loop. It is typically easier to implement a looping instruction than it is to evolve the equivalent out of mathematical, jump, and conditional instructions. In the latter situation, many specific, ordered mutations would be required to implement a basic loop, including the ability to terminate this loop at the appropriate time. By directly implementing these concepts in a language, it essentially narrows the search space, which is often beneficial to specialized questions.

Another advantage of a genetic language is that implementing language constructs can increase or decrease the impact of a mutation on certain aspects of the morphology. This is possible due to instruction complexity in which an instruction pleiotropically affects more than one trait or otherwise accomplishes multiple atomic actions. The complexity of a mutation to one of these instructions can be characterized as a single atomic instruction being replaced by a series of other atomic instructions. Thus a single mutation to or from a complex instruction can have a huge effect because that mutation would simultaneously impact the entire series of instructions that the single instruction represents. While this can be a disadvantage due to the more dramatic (and hence more often detrimental) nature of mutations, it also enables complex concepts to be represented in individuals—something that would otherwise occur quite rarely. Another

disadvantage is that adding these more complex instructions, I either bloat the set of possible instructions (making a mutation to any one of them much less likely) or else increase the number of mutations required to make small changes. However, a properly designed language can minimize (but never eliminate) these tradeoffs. For example, in a morphological system it may take many mutations to change a specific parameter for a single component of the phenotype already in existence, but it would take only one mutation to add an entirely new component with default parameter values.

The underlying genetic language is a critical topic for most of the systems that evolve both morphology and behavior. A common thread among them is the focus on symmetrical, or recursive, construction of body parts (Sims, 1994a; Komosinski and Rotaru-Varga, 2001b; Hornby and Pollack, 2002). Some of the research done by Hornby and Pollack (2002) as well as Komosinski and Rotaru-Varga (2001b) has shown improvement in both speed and reliability when evolving movement using a recursive language over a direct encoding in certain situations. Hornby and Pollack (2001b) also observed an order of magnitude increase in body parts and a higher degree of regularity. Komosinski and Rotaru-Varga also noticed more ordered structures including modularity and segmentation. These experiments are important in building an experimental framework for these types of systems.

However, the problems associated with direct encodings are poorly understood. Why do they seem to perform so poorly compared to recursive languages? Is it because a direct encoding is too complex, requiring a massive number of mutations to fine tune component parameters? Will the same results occur between symmetric versus non-symmetric languages both of which are not direct encodings? These questions remain

exciting topics in the field, but for the most part researchers have moved on to more dynamic languages. I developed a language for WhirlingDervish that was inspired by a preexisting genetic language from the Avida (Ofria, 2004) system.

The Avida system strives to answer both biological and general evolutionary questions through the use of self-replicating computer programs (as opposed to 3D evolving constructs). The most common type of evolutionary goal for these individuals is to perform math and logic functions. Any of these functions can be solved because Avida employs the use of virtual CPUs with a Turing complete pseudo-assembly language. In addition, the Avida language provides the necessary instructions that allow the individuals to self-replicate. The WhirlingDervish language uses some of the mathematical functionality of the Avida language including the use of stacks. However, the concept of self-replication is not used and the respective Avida instructions are replaced by instructions that build and alter a three dimensional virtual body. The other primary difference is that the Avida genome is circular while in WhirlingDervish they are a linear array of instructions. On the other hand, both languages are executed in order to produce their phenotypes. For more details on the language used here, see Appendix A.

2.3.3 Replication and Mutation

Once all of the individuals have been simulated and their fitness calculated, it is time to replicate the individuals and generate a new population for the next generation. The process I use is called fitness proportional selection. First each individual's fitness is divided by the total fitness for the whole population. This essentially creates a weight for each individual that represents it percentage chance of replication during a single round of selection. Then, for each empty slot in the new population, except one, I generate a

random number that corresponds to a specific individual. The chosen individual is replicated then put backing into its population, sampling with replacement. The extra slot in the new population is reserved for the most-fit individual in the current population who gets a single exact copy of itself generated and placed into the new population, making this a system with an elitism of one.

When a wire-frame individual has been selected for replication, the process of genome duplication begins. As each instruction is copied from the parent genome to the offspring's genome there is a chance for mutation (P). However, crossover was not used in this system because the added complexity was not required to accomplish the goal. The chance of mutation is calculated at the beginning of the copy process and remains constant throughout. The reason the chance of mutation is calculated each time is that the value is composed of two components: one that is fixed and applies to every individual throughout the evolution process (P_n), and one that is based on the length (ℓ_g) of the parent genome being copied (P_g). The chance of mutation is given in the following equation:

$$P = P_n + \frac{P_g}{\left(\ell_g + 1\right)} \tag{2.16}$$

The addition of one to the length of the genome exists due to the fact that some genomes may be empty, as is the case for the initial individuals. This equation has many potential uses, but I only use P_g for the mutation rate with a value of 1.0, by setting a value of 0.0 for P_n. Both P_n and P_g can be set in a configuration file and will remain unchanged throughout the run.

If a mutation occurs while copying an instruction, one of four possible types of mutation is used to determine what will happen to the copied instruction. The four types of mutation are point mutation, insertion, deletion, and instruction duplication. Each mutation type has a probability associated with it that is configurable and must sum to 1.0. The probabilities used in this thesis are shown in Table 2.3.

Table 2.3. Mutation rates

insertion	0.05
duplication	0.05
deletion	0.1
point	0.8

The effect of insertions, duplications, and deletions are simple. When an insertion occurs, an extra random instruction is added to the offspring's genome immediately after the most recently copied instruction. To allow insertions at the beginning of the genome, there is always a chance of insertion before the first instruction is copied. Duplication is essentially an insert except the parent instruction is copied twice. Deletion simply does not add a copy of the parent instruction into the offspring's genome.

There are two types of point mutation: random instruction and parameter modification, with a 50% chance that it will be one or the other. The random instruction disregards the parent instruction and creates a new random instruction with a random parameter. The parameter modification works differently if the current parameter is a name rather than an explicit vector. If the parameter is a name, a new random parameter is generated with a 65% chance of becoming a random explicit vector. If the current parameter is an explicit vector there is still a 65% chance of getting a new random explicit vector. Otherwise, a random component of the vector (x) is chosen and a modifier is added to it as shown in the following equation

$$x = x + 1.1^{random\_number(-1,1)} \quad (2.17)$$

The value of x is capped between 1 and -1. Also, the chance that a parameter will change into a random explicit vector is configurable, and the choice of 0.65 was chosen to give a slight bias to explicit vectors.

2.4 Runtime Abilities

An important attribute of WhirlingDervish is its ability to evolve large populations in real-time. The success stems primarily from the simplicity of the wire-frame individuals and the environment. The simplicity gives a large reduction in the time spent modeling individuals through simulation, where the simulation consumes the majority of a run's overall computational time. However, other researchers have also achieved large population sizes. One such researcher is Ventrella (1994; 1998) who simplified his constructs to two dimensions in order to achieve population sizes of up to 200 individuals in real time on an SGI Indigo2 machine in 1996. His constructs were motorized stick figures that had to evolve the ability to swim within a fixed size circular pond in order to survive. Another is Hornby and Pollack (2001b) who were able to have a large population of complex individuals made up of many components that consisted of bars and joints controlled by a neural network. The bars and joints contribute structurally and have corresponding motorized capabilities, which is controlled by a neural network. The experiments were run with 100 individuals, each with an upper limit of 350 bars, for up to 500 generations. In another similar experiment, they used a population size of 200 individuals for 250 generations (Hornby and Pollack, 2002). Bongard and Pfeifer (2003) used the Artificial Ontogeny system and were able to have a population of size 200.

In WhirlingDervish, the average time taken to evolve a population of 500 wire-frame individuals for 500 generations is approximately 25.8 hours on a single 2.8 Mhz Athlon processor, or about 1.548 times real-time. The body size for a wire-frame individual can vary greatly since the body size begins at zero and has no explicit upper limit. However, the size is somewhat constrained by only giving a wire-frame individual an increase in fitness for up to 10 point-masses and penalizing fitness when the number of springs exceed 54 (one less than the number of springs in a complete graph with 11 points). The average runtime was taken using the default genetic language, a two minute simulation time, and a time step of 0.01 seconds over fifty replicate runs. This speed is at the high end relative to the other systems for evolving morphology and mobility. However, it is difficult to directly compare speeds due to the disparate body sizes of individuals evolved in WhirlingDervish, the difference in the types of individuals evolved, and the information reported about their configuration.

Chapter 3 Analyzing Convergence and Stagnation Through Diversity and Variation Measures

Current research of diversity measures in evolutionary computation suggests possible, beneficial methods for characterizing convergence and stagnation in 3D morphology and behavior evolutionary systems. This research commonly examines the change in genetic and phenotypic diversity within a population over time. Practical motivations often involve controlling diversity through the use of operators that utilize active feedback of the current diversity. Conceptual motivations strive for a better understanding of the underlying dynamics of evolutionary algorithms, their structures, methods, and operators. My motivation is to find a way to quantify the ability of the WhirlingDervish system to continually evolve or improve solutions. Studies of other evolutionary aspects can then use this quantity as a reference point for comparisons. Therefore, this chapter will examine how informative the various diversity measures are towards quantifying convergence and stagnation.

To make use of diversity measures both a method for classifying individuals, grouping similar individuals, and deciding on the unit of classification are required. The best choice of classification unit is phenotypic behavior, as it is the overarching goal of the system. However, with the behavioral complexity of individuals in WhirlingDervish, where the fitness scheme only approximates the desired outcome, I am unaware of computational models capable of classifying them. Currently, the majority of the classification occurs by visual analysis, which also makes it difficult to judge the success of a system in terms of desired solutions. Therefore, I rely on quantitative aspects of the individuals such as phenotypic traits: fitness and the xz-distance an individual travels.

Some work already exists for classifying phenotypes in 3D morphology and behavior evolutionary systems (Komosinski et. al., 2001a). This work discarded classification based on genotype information. Instead, a similarity measure is used based on discrete structural characteristics of the phenotype such as the degree at each vertex. While their method may do well at classifying their structure, it does not address the classification of behavior. Also, individuals in the WhirlingDervish system may express different structures over time even though the number of connections at a vertex remains constant. To this end, their measure would not work well at classifying structural or phenotypic behavior of individuals in WhirlingDervish, but may form the foundation for future computational models.

Section 3.1 presents an overview of diversity measures taken from other research studies in evolutionary computation and population biology. At that point, I will also examine the differences between their current usage and any modifications I made in order for them to work with WhirlingDervish. While highlighting these differences, I will also discuss how the interpretation of those measures needs to be adjusted from the problems they were tested on originally, and their potential application to open-ended evolutionary systems. Section 3.2 will then provide a detailed description of all of the diversity measures used in this thesis. Next, the experimental design, in Section 3.3, describes the configuration and important parameters for the WhirlingDervish system to generate fifty replicate runs used as data for this chapter. Then, Section 3.4 provides an analysis of the application of the diversity measures to those runs, and the conclusions follow in Section 3.5.

3.1 Definitions and Methodology

3.1.1 Definitions

Diversity is a measure that describes a population in terms of classes, or groups, taking into account both how many classes there are and how evenly those classes are distributed within the population. The term **class** represents the classification of individuals into groups based on measures of similarity. The classification of individuals divides the entire population such that every individual falls into a single class. Thus each class represents a proportion of the population. **Evenness** is a measure of how close to equal the proportion for each class is to the others. Evenness is maximized when all of the classes contribute the same number of individuals to the population. In biology, it is often difficult to determine the proportions for classes, or species, so when the evenness information is unavailable, richness is used. **Richness** is a raw measure of diversity that only represents the number of different classes.

In evolutionary computation and population biology, **convergence** is the loss of genetic diversity. Convergence for genetic algorithms occurs when the genomes are similar enough that crossover can only produce a limited range of offspring all of which have equal or lesser fitness (Fogel, 1994). Studies conducted in genetic programming (McPhee and Hopper, 1999; Daida et. al., 2004) have shown that using only crossover and randomly generated initial populations will cause rapid convergence through the loss of genetic material, which can never be replaced or recovered. Primarily, convergence occurs in optimization systems where the selective pressures and environment are held constant. Other aspects of an evolutionary system that can contribute to convergence are

exponential fitness growth and low mutation rates. However, convergence is often a desired quality for many of the problems addressed through evolutionary computation.

While convergence is often used as an indicator that a final solution has been reached, many research studies have been conducted on how to prevent premature convergence. **Premature convergence** is the convergence on a suboptimal solution for a problem that has multiple solutions where some are suboptimal. Once premature convergence has occurred, the population has usually lost much of its genetic diversity and is unable to find an optimal solution. In an attempt to prevent premature convergence, the study of diversity has become an important topic, particularly when focusing on controlling diversity (Burke et. al., 2004).

Stagnation occurs when a population has not progressed toward its evolutionary goals (approximated here by fitness) for a period of time. It is independent of the genetic variation and does not preclude beneficial mutations within reach of the population's search capability. However, the presence of such mutations does not guarantee that a population will find them in any finite amount of time (Bossert, 1967). Two interrelated issues with stagnation are determining why a population has stagnated and the trade-off between continuing to search for improvements and stopping the search. In systems where the fitness landscape is unknown or complex, it may not be possible to determine the cause of stagnation, but the state is still quantifiable. However, the existence of stagnation does not imply that a better solution would not be found later in the evolutionary process.

3.1.2 Population Diversity

Given all of the different diversity measures available, I settled on four primary methods for calculating the diversity of a sampling unit, or population. They are richness, entropy, Simpsons's diversity index, and McIntosh's diversity index taken from Magurran (Magurran, 2004). The Simpsons's and McIntosh's diversity indices are often defined in terms of ecological diversity, incorporating both evenness and quantity of unique species. While these measures deal with species, it is easy to convert them to different units such as genotype and phenotype diversity within a population. All that is required is to define a sampling unit such as a population and a method for classifying individuals into distinct groups within that population.

The use of class richness is common in evolutionary computation as noted in (Burke et. al., 2004). It is also called variety when in reference to unique genotypes (Koza, 1992). Class richness is a raw measure of diversity, and only takes into account the quantity of classes, not the distribution. Lacking that key bit of information about evenness limits the information content of this measure. However, there are situations where the richness measure will be a good approximation of the other measures, which occurs when a high percentage of unique classes exist in a population. When the percentage of unique classes is low, it is impossible to determine their distribution within the population based solely on the count.

While most uses of richness are defined in terms of genotypes, it is also used in genetic programming for phenotypic diversity (Rosca, 1995b). In this situation, phenotypic classes are grouped by unique fitness values. Rosca also suggested a coarser method for classifying fitness by binning fitness values based on a range of values rather

than an exact match. This alternative method was tested in WhirlingDervish because of the specialized "Jitter" function. Jitter modifies each component of the phenotype by a very small delta causing differing fitness values for multiple measures of the same individual. However, the use of the alternative method was dismissed when it did not show additional information over the non-binned version.

While the use of entropy as a diversity measure was found in terms of ecology (Magurran, 2004), it has also been used in evolutionary computation and artificial life (Ray, 1994; Adami, 1998). In these systems, the method for calculating diversity is the same as the ecological method, but is calculated in terms of either genotypes or phenotypes (Rosca, 1995a). Essentially, entropy is a measure that takes into account both the quantity of classes and their distributions. Maximal entropy occurs for a population with a given number of classes when all of those classes fill the same proportion of the population (i.e., there is maximal evenness). Ray's (1994) use of a genetic entropy measure is a good example of what can be learned by using an entropy measure. In his system, Ray was able to show that sharp drops in entropy corresponded to the introduction of new innovations, which swept the population.

3.1.3 Variation

There are many different measures in genetic programming that are labeled "diversity". However, many of these measures use statistics at the level of individual comparisons. To help distinguish between these and the diversity measures mentioned in Section 3.1.2, I am going to call them variation measures. There are four categories of variation used in this thesis: phenotypic, genetic, inferiority, and information relating to the most recent common ancestor (MRCA). These categories use measurements that

move away from looking at classes and their abundance within a population and focus on comparisons between individuals.

The first variation category utilizes measurements of phenotypic traits. Since there is no good way of quantifying behavior, I will focus on fitness and morphological traits. The morphological traits I use include height, xz-distance moved, number of point-masses, and number of springs. These traits will be visualized in two ways: as an overall average of that trait across an entire population over time and how those traits change along a lineage. The changes in the five phenotypic traits should help distinguish convergence and stagnation factors.

The second category is genetic variation, which is based on edit distance. This category is broken into two measures intended to quantify the relationship between two genomes. The first measure quantifies how dissimilar are two genomes while the second measure quantifies the amount of similarity as the proportion of shared genetic material. Both measures are based on the length of the two genomes being compared and the edit distance between them. Also, the relationship between the two measures is that the similarity equals one minus the dissimilarity when both genome lengths are equal. The specific formulas are given in Section 3.2.

The use of edit distance itself is commonly used in evolutionary algorithms to determine genetic diversity and variation. In genetic algorithms, Hamming distance is used because each genome is often a fixed length linear array of bits. Genetic programming had to develop specialized edit distance formulas due to the use of a tree style genome representation. According to (Burke et. al., 2004), the two most common edit distance measurements in genetic programming are 1) the number of changes

required to transform one tree into another and 2) weighted edit distance (Ekart and Nemeth, 2002). The second method first fills in each tree with empty nodes to make it full, then computes the edit distance to be a weighted sum of the Boolean difference at each node, where the root has the most weight.

My method for calculating edit distance is Levenstein distance. The basic algorithm calculates distance as the sum of the number of mutations, insertions, and deletions required to go from one string to another. In this situation, the genome is the string and the instructions are the characters. To simplify edit distance calculations, I consider sites from two genomes identical only if both instructions and arguments are a perfect match.

The inferiority measure comes from artificial life. Adami (1998) created this measure as an analog to energy in a physical system, where the population has a pressure to reduce its average inferiority to zero, called the ground state. This idea represents an overall trend for the population to converge. Originally, inferiority was defined in terms of an individual's replication rate, but it was trivial to convert it to using fitness. Here, inferiority for an individual is the difference between the best fitness in the population and the fitness of that individual. Thus, the average inferiority is the average difference in fitness for the population.

The final variation category contains three measures that utilize the most recent common ancestor (MRCA). The first measure is simply the generation of the MRCA for the final population of a given run. In some sense, this value is an estimate of the phylogenetic diversity, but there is a potential confounding effect caused by outlier individuals within the tree. These individuals diverge early relative to the rest of the

individuals in the population. This can cause a skew in the MRCA generation when the majority of the individuals are closely related. Even with this confounding effect, this measure can still provide useful information. Intuitively, the closer in generational time the MRCA is to the last generation, the more likely the genetic variation will be low since there would be little time to diverge. Likewise, the further back in time the value, the more likely the final population has a diverse set of genotypes. However, if the time difference between the final generation and the generation of the MRCA grows too large, it can indicate low selection pressure, or even no selection.

The other MRCA measures are averages that use the similarity measures from the genetic variation category. Application of those similarity measures occurs between the MRCA and each individual in the final population. The average dissimilarity measure approximates the level of divergence between the MRCA and the final population as the average proportional difference. This measure is an estimator for the change in variation over time, which in turn can be used to approximate the likelihood for convergence. The average similarity measure approximates the quantity of genetic material from the MRCA conserved in the final population. This measure can potentially indicate if a new phenotype has arisen during a run. It also has the potential to indicate that the population is incrementally improving on some phenotypic aspect represented by the conserved genetic material.

The idea of using the average similarity measures with the MRCA was inspired by the work of McPhee and Hopper (1999), which was later rechecked Daida et. al. (2004). Their work examined how much genetic material from the initial population contributes to the final population in a genetic programming system with crossover but

no mutation. This experiment showed that all else being equal, the situation forces severe loss of diversity in both genetic material and genetic structure. The reason for this was that crossover could not introduce new genetic material, and when something was lost, it could never be regained.

3.2 Diversity and Variation Measures

The following is a short description of each diversity and variation measure accompanied by a formula when appropriate.

Population Diversity Measures:

- Class Richness- It is simply the count of the number of unique classes in a population.
- Shannon Diversity (Entropy)- Measure of disorder in the population, where N is the population size and N_j is the number of individuals in class j.

$$-\sum_j \frac{N_j}{N} \log_N \frac{N_j}{N} \tag{3.1}$$

- Simpsons's Diversity Index - A measure that accentuates uneven class proportions in a population by weighting each as the square of its abundance, where N is the population size and N_j is the number of individuals in class j.

$$1-\sum_j \frac{N_j\left(N_j-1\right)}{N(N-1)} \tag{3.2}$$

- McIntosh's Diversity Index - A measure that treats the proportion for each class as a position in n-dimensional space, where n is the number of classes. The position for a given class is only along its own axis at a distance from the origin equivalent to its proportion in the population. The measure of diversity is then calculated as the average distance. N is the population size and N_j is the number of individuals in class j.

$$\frac{N - \sqrt{\sum_j N_j^2}}{N - \sqrt{N}} \tag{3.3}$$

Variation Measures:

- Average phenotypic trait over time - The sum of the trait value for a given population divided by the population size.
- Similarity- The proportion of genetic material in the smaller genome that also appears in the larger one, where L is genome length and D is the Levenstein distance.

$$\frac{L_{\max} - D}{L_{\min}} \tag{3.4}$$

- Dissimilarity- The ratio of the differences between genomes to the length of the longer genome, where L is genome length and D is the Levenstein distance.

$$\frac{D}{L_{\max}} \tag{3.5}$$

- Inferiority- The difference between the most fitness individual and the i^{th} individual in a given population.

$$\varepsilon_{\max} - \varepsilon_i \tag{3.6}$$

- MRCA generation- The generation of the MRCA for the final population.
- MRCA edit distance- The average edit distance between the MRCA and each individual from the final population.

3.3 Experimental Design

To accomplish the goals of this chapter, fifty replicate runs were generated using the WhirlingDervish system. At the completion of those runs, all of the data were processed and the necessary information was extracted. Section 3.4 discusses the results from the application of the diversity and variation measures to the extracted data. However, to better understand the results, this section provides an overview of the configuration and important aspects of the WhirlingDervish system.

The configuration uses a fixed population size of 500 individuals. With the initial population consisting entirely of empty genomes, the run continues for 499 generations. At the end of each generation an entirely new population is created using fitness proportional selection with an elitism of one. Once an individual is selected to generate an offspring, its genome is copied instruction by instruction into the genome of the offspring. This process occurs external to the individual; the individuals do not technically self-replicate. As each instruction is copied into the offspring's genome, there is a chance mutations will occur based on a rate computed prior to the copy process. That rate is genome length dependent and set such that there is an average of one mutation per

offspring. The four types of possible mutations are point, insertion, duplication, and deletion. No crossover is currently used in WhirlingDervish.

The evolutionary goal of the WhirlingDervish system is to produce mobile individuals that move consistently over time. Approximating this goal, a simply additive fitness function defines selective pressures using the following measurable phenotypic traits: (1) the xz-distance between the original center of mass and the final center of mass, (2) the number of point-masses, (3) the vertical distance from an individuals lowest point to its highest point, and (4) the height from the lowest point to the center of mass. Both the xz-distance moved and the height to the center of mass are measured in meters after a two-minute evaluation period and scaled by a factor of 3, and the number of points is scaled by the value 0.0254. Also, a few of the primary constraints consist of a cap on the fitness benefit from the number of point-masses at 10 and a cap on the two height traits at 15 meters, but these constraints do not restrict any trait from exceeding their caps.

The fitness values generated during a run must be recomputed prior to analysis. The primary reason that I compute the fitness again is to remove the stochastic perturbations that are normally applied to each point-mass in a newly generated individual before its evaluation period. These perturbations normally cause multiple fitness measurements to vary widely for the same individual. Thus, without these perturbations, the true fitness value of an individual is computed and could be used for comparisons across runs. The other reason to recompute fitness is to achieve a consistent fitness measure that will correlate better with visual analysis since the specific perturbations used during the experiment are difficult to duplicate and are not applied.

3.4 Results and Discussion

This experiment examines the occurrence and likelihood of convergence and stagnation in the WhirlingDervish system under the conditions described in Section 3.3. This section will first examine convergence by exploring the results of different diversity and variation measures. The same will then occur for stagnation, which will draw on some of the measures and conclusions for found for convergence. However, to be clear, the results and conclusions drawn from this experiment only apply to the limitations given in the design. It is entirely possible that given a long enough period of evolutionary time, the results may differ. At a future time, the procedures used to ascertain results for this experiment will be conducted on longer runs and multiple data sets for verification.

3.4.1 Convergence

This section utilizes genetic diversity and variation measures in order to examine convergence. In the process, I explore the selection pressures within the system and their impact on the assertions drawn from the various measures.

3.4.1.1 Genetic Diversity

To recognize convergence in a system, the first step is to examine its genetic diversity. Figure 3.1 shows the genetic diversity as measured by the four diversity measures described above in Section 3.2.

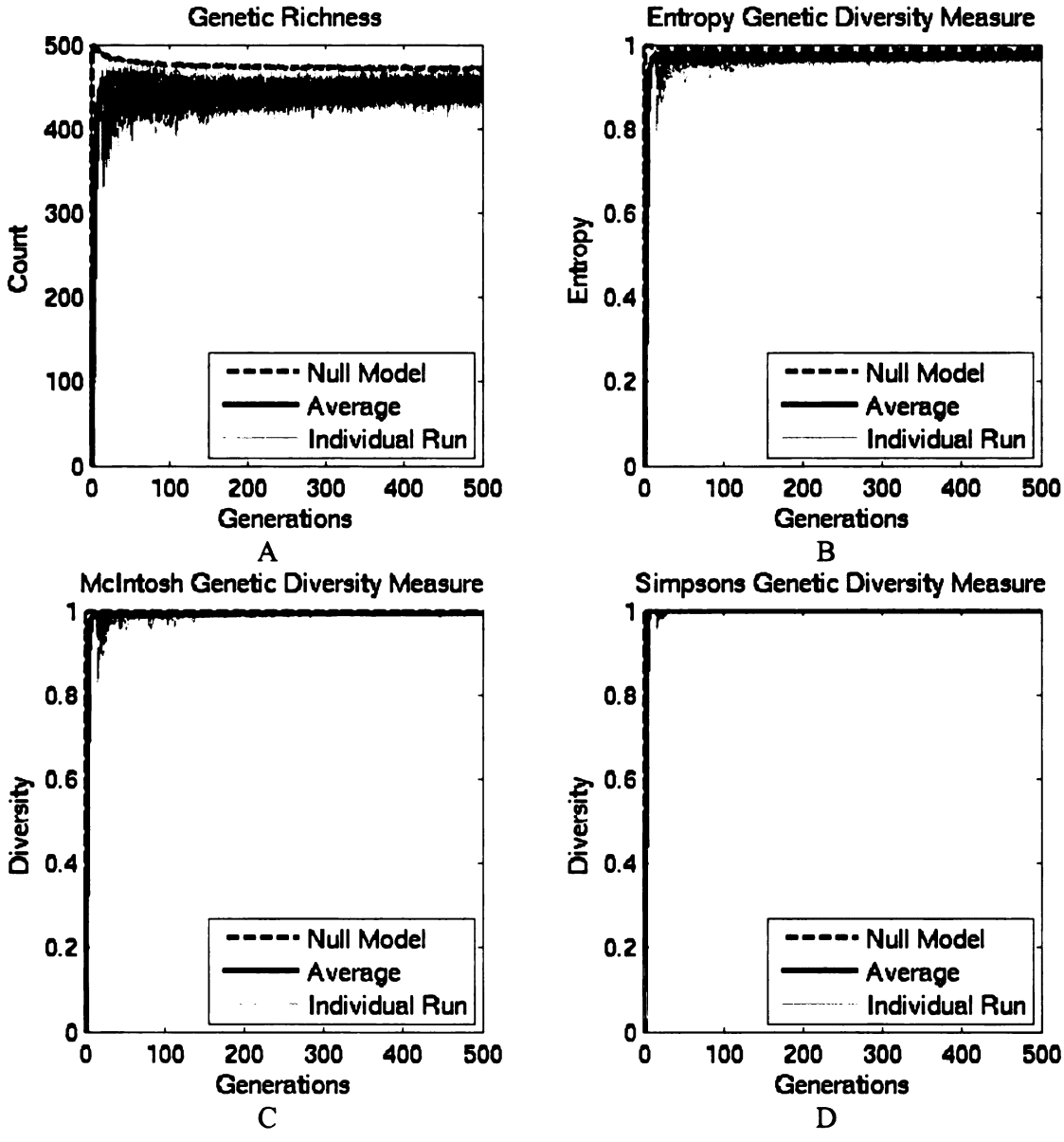
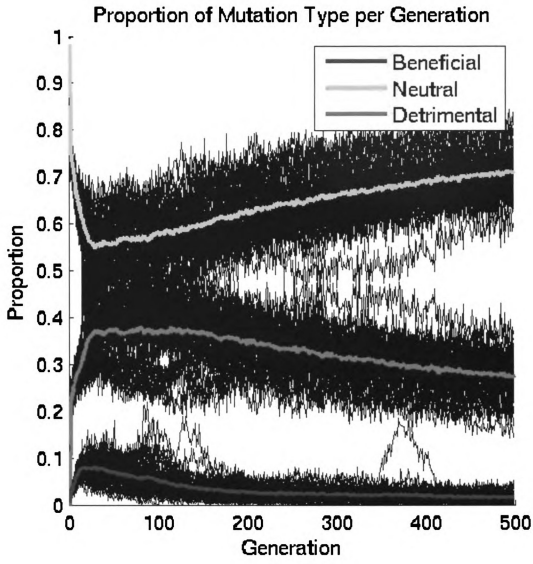


Figure 3.1. Genetic diversity using the following measures (A) richness, (B) entropy, (C) McIntosh's diversity index, (D) Simpsons's diversity index. Each graph contains 50 grey lines, where one line represents the diversity over time for a single run. The thick black line is the average diversity across all of the runs, and the thick dashed line represents the expected value given by a null model where no selection is present.

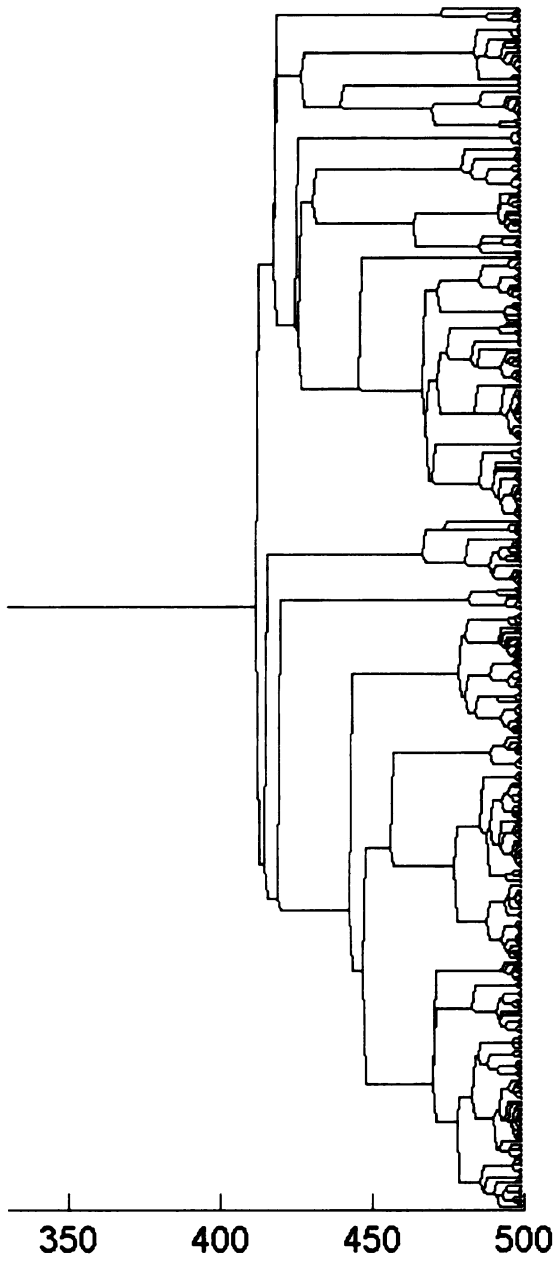
The graphs in Figure 3.1 clearly show a high level of genetic diversity after an initial loss and recovery near the beginning. The McIntosh and Simpsons's diversity indices show a maximal diversity over the majority of the run, while the richness and entropy measures

show a high and slightly increasing diversity. Therefore, these graphs strongly indicate a lack of genetic convergence.

In an attempt to understand why the genetic diversity is so high, a null model was created to estimate he number of unique genotypes that would occur in the absence of selection. These values are represented as a thick, black dashed-line in Figure 3.1. While it is interesting that the genetic diversity is high for most of the run, it appears to be close to the null model estimate. This suggests a weak selective pressure, most likely brought about by the potential accumulation of a high number of neutral mutations. Figure 3.2A shows the percentage of neutral, detrimental, and beneficial mutations at each generation. The weak selective pressure also means that it is hard to weed out individuals and sweeps will occur infrequently, if at all. It may also lead to a high distribution of groups within the population that have approximately equivalent fitness, and are fairly unrelated within a phylogenetic tree. Figure 3.2B is an example of phylogenetic tree.


Proportion of Mutation Type per Generation
Beneficial
Neutral
Detrimental
Proportion
Generation
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
100
200
300
400
500

A



B

Figure 3.2. (A) The proportional representation of neutral, beneficial, and detrimental mutations for each generation. (B) The end of a phylogenetic tree including the most recent common ancestor of the final population. This particular tree has a later than average most recent common ancestor and was chosen to show that even a tree this young has many groupings containing long branch lengths.

In view of the lack of convergence and the weak selection, it is important to understand some of the mechanisms underlying the evolutionary process. First, all of the

genomes in the first generation are empty, which means that the second generation is always comprised of genomes with at most one instruction. There is only an 8% chance that a non-neutral insertion will occur, and all of the individuals with non-zero fitness will have the same fitness value at this point in time. Once the population has a mixture of zero and non-zero fit individuals, the individuals with zero fitness have no chance of reproducing, thus all of the non-zero fit individuals will be the only progenitors. However, because fitness proportional selection is in use, all of the individuals with non-zero fitness will have the same chance at reproduction.

This process will continue for the first few generations until the evolution of a stable individual with height, which should require at least four point-masses to not fall over. The first individuals that retain height will begin to sweep the population, which is noticeable in Figure 3.1. However, the sweep is weakened as successful offspring are now in competition with only small relative differences in fitness between them, which will reduce selective pressure again. It is understandable that the transition from receiving fitness benefit from only the number of point-masses, which is small, to individuals with height would have the largest fitness differential and hence the most likely to trigger a sweep. The fitness gains from the other components of fitness are gradual and provide a more constant, progressive improvement, which dramatically reduces the likelihood of a full sweep. The term sweep here refers to a single genotype rapidly filling the entire population with its immediate descendents because of its high fitness relative to the other individuals.

3.4.1.2 Genetic Variation

Genetic variation has the potential to strengthen the usefulness of diversity measures by quantifying the difference between classes within a population. This section examines how the use of genetic variation contributes towards characterizing convergence. I begin by examining the average similarity and dissimilarity between all pairs of genomes for each population over time (Figure 3.3).

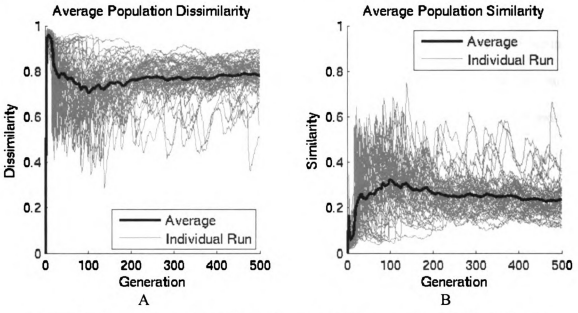


Figure 3.3. (A) Average dissimilarity over time (B) Average similarity over time

In Figure 3.3, there is rapid variation in average similarity and dissimilarity, particularly during the early generations. While some runs exhibit large deviations from the average across populations, most do stay near that average. It is also interesting that the initial rapid change in the average accompanies the genotype explosion mentioned above for genetic diversity. As a sweep begins wiping out radically different genotypes and replacing it with offspring, the dissimilarity drops, and the average shared genetic material increases, as shown in the first 50 generations. Interestingly, the averages

continue their trend for another 50-100 generations after the spike representing the loss of genetic diversity.

I suspect the continued trend in the averages is due to slight domination, however random, of a few phylogenetically related groups before the groups have grown in size enough to ensure their representation in the next generation. One of the most useful pieces of information is continual increase of dissimilarity, and decrease in similarity, for the rest of the run after the initial, respective thrusts. These continuous trends support the high genetic diversity found above. However, with the knowledge that the neutral mutations increasingly dominate over the other mutation types, these trends are most likely caused by the accumulation of neutral material in the genomes. A way of testing the assertion that genomes are accumulating neutral mutation, and potentially improving the significance of both measures, would be the removal of all neutral instructions before the measures are applied. However, that test is beyond the scope of this experiment due to the computational expense.

The other issue with Figure 3.3 is that the average across runs of the average dissimilarity, and similarity, do not show the oscillations apparent in some of the individual runs. In order to better understand what these oscillations represent, I examine a single run with oscillations in Figure 3.4, which shows average fitness and average dissimilarity over time. What is interesting about this figure is that each time fitness increases, dissimilarity briefly drops, presumably due to a hitchhiking effect as the new best solution takes over the population.

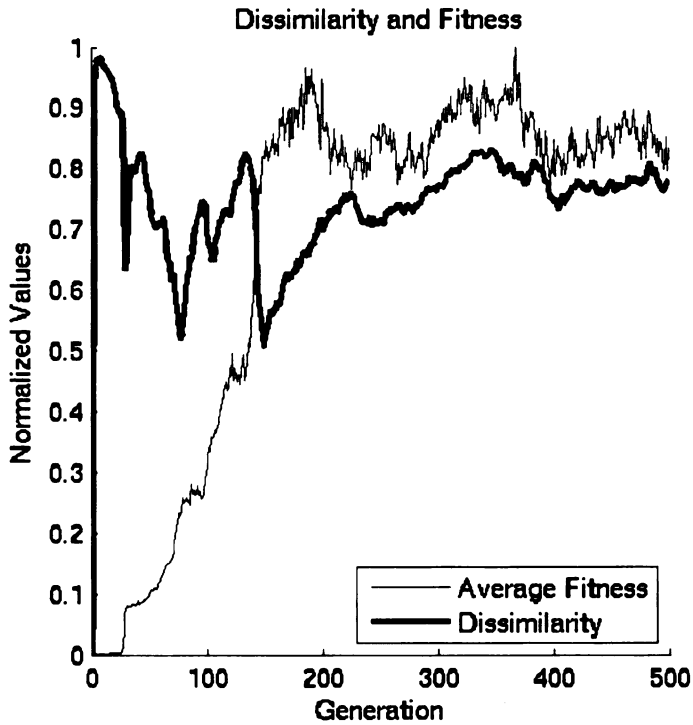
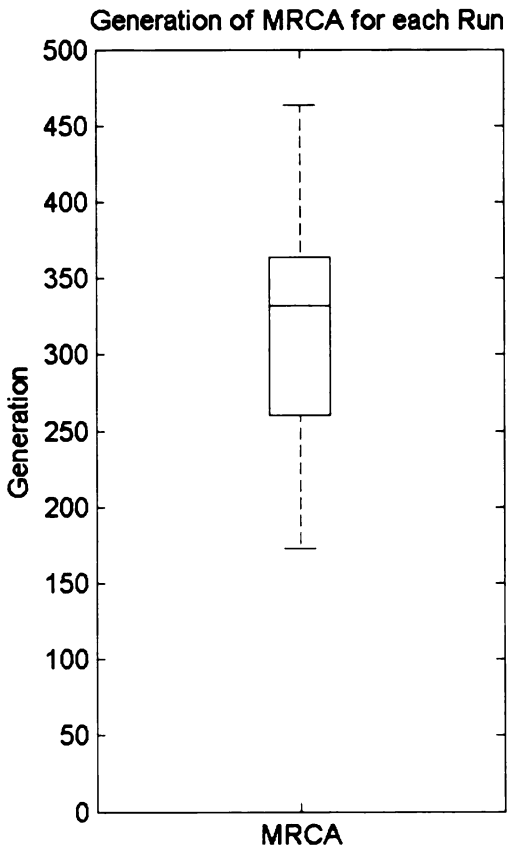


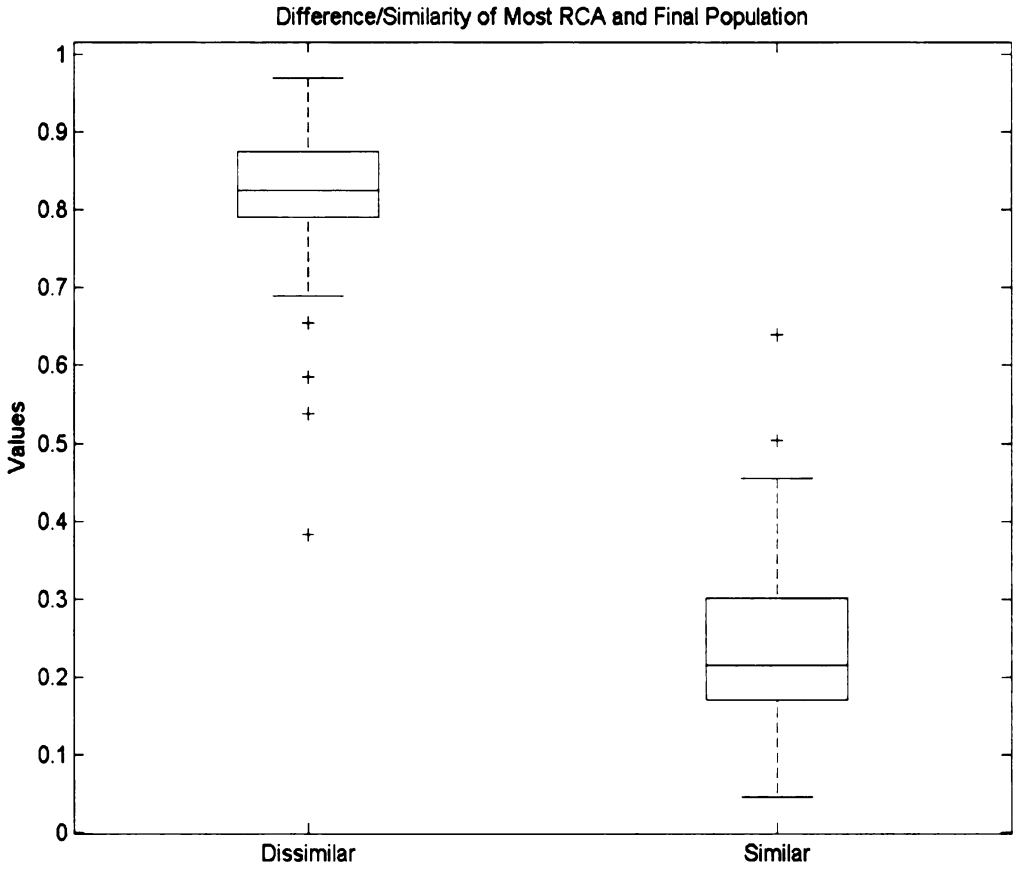
Figure 3.4. Average dissimilarity and average fitness over time.

3.4.1.3 MRCA Variation

This section examines the use MRCA variation towards characterizing convergence. The following graphs (Figure 3.5) show results of the variation measures utilizing the MRCA.


Generation of MRCA for each Run
500
450
400
350
300
250
200
150
100
50
0
Generation
MRCA

A



B

Figure 3.5. (A) A boxplot depicting the generation of the most recent common ancestor for each run. (B) A boxplot depicting aggregate Dissimilarity/Similarity between the most recent common ancestor and all of the individuals in the final population for each run.

It is apparent from Figure 3.5A that the average MRCA occurs at approximately the 350^{th} generation, which is 70% of the total run time. Using this value as a rough indicator of diversity, suggests a moderately diverse final population. Figure 3.2B showed a phylogenetic tree whose MRCA occurs at the later end of the spectrum suggesting that if a tree that late in time has such a diverse grouping of leaves, then it is reasonable to expect the ones who are earlier would also be diverse. I verified this

assertion by visually examining the phylogenetic tree for each run. The other interesting item about the MRCA generation data is that the earlier half of the generation times falls within 40-70% of the total run time. These earlier generation times may indicate a low selective pressure because of a lack of domination from any given group of individuals, which agrees with the results shown above in Section 3.4.1.1.

When examining the dissimilarity of the MRCA individual and the individuals from the final population, the average dissimilarity is slightly higher than 80%. The increase of genetic dissimilarity can arise due to a combination of genetic drift and the insertion of new genetic material. The data do not distinguish between the two methods that increase dissimilarity, so without extraneous information, the average dissimilarity indicates a low likelihood of genetic convergence. The few outliers with low dissimilarity show that at least some of the runs are likely to have converged. However, with the low selection pressure established earlier, it is unclear how much the converged runs dominate the results the dissimilarity results of Figure 3.3A.

The similarity measure shown in Figure 3.5B shows that on average about 20% of the genetic material is conserved from the MRCA individual. I conclude that some kernel of genetic material is being conserved and the continued increase in fitness occurs by both tweaking and building off of this kernel. It is also not possible to tell the relative importance of the conserved material versus the recently generated material in regards to the phenotype. Thus, the difference in proportion of conserved material between runs could relate to the variable amount of genetic material to generate their different phenotypes.

3.4.2 Stagnation

This section utilizes phenotypic variation measures and fitness distributions in order to examine stagnation.

3.4.2.1 Fitness Improvement and Morphological Convergence

One simple way of monitoring an evolving population is to examine the change over time of various phenotypic traits. This can be visualized in two different ways. The first is to average the trait across an entire population. The second involves finding a representative individual at each generation. Two common methods for choosing a representative are selection of the most-fit individual from the population at each generation, or selection of an individual from the lineage that led to an individual (typically the most-fit) in the final population. Even though using the most-fit individual is more common in evolutionary computation, I chose to select individuals from a lineage because these data are less noisy. Figure 3.6 visualizes the noise difference between the most-fit individuals and lineage individuals in terms of height. This noise is also evident in all of the other phenotypic traits, though less extreme in fitness and distance moved data.

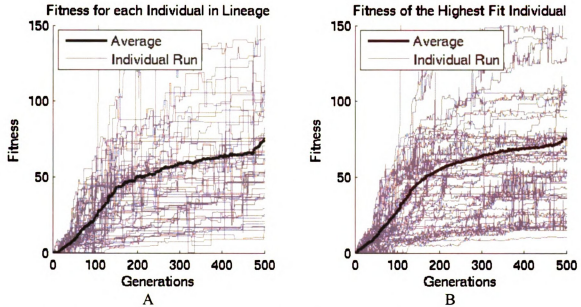
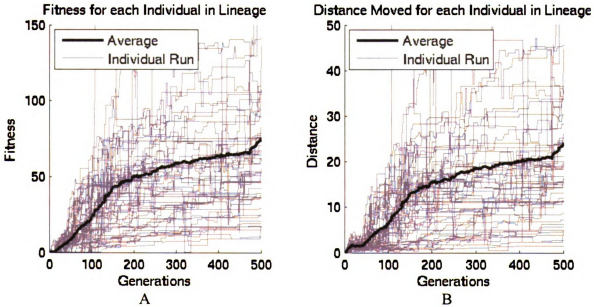


Figure 3.6. Noise difference between data defined by (A) lineage and (B) most-fit individuals per generation.

Two sets of visualizations follow that present all five phenotypic traits over time along a lineage (Figure 3.7) and averaged across the population (Figure 3.8).



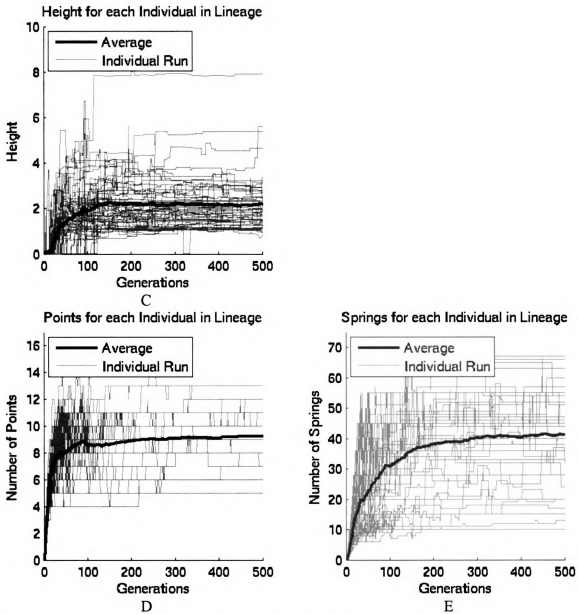
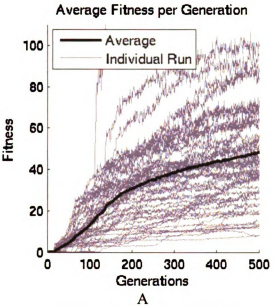
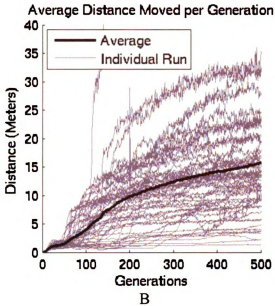


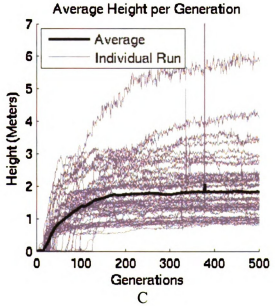
Figure 3.7. Phenotypic trait changes along a lineage for the traits (A) fitness, (B) xz-distance moved, (C) height, (D) number of point, and (E) number of springs.


Average Fitness per Generation
Average
Individual Run
Fitness
Generations
0
20
40
60
80
100
0
100
200
300
400
500

A


Average Distance Moved per Generation
Average
Individual Run
Distance (Meters)
Generations
0
5
10
15
20
25
30
35
40
0
100
200
300
400
500

B


Average Height per Generation
Average
Individual Run
Height (Meters)
Generations
0
1
2
3
4
5
6
7
0
100
200
300
400
500

C

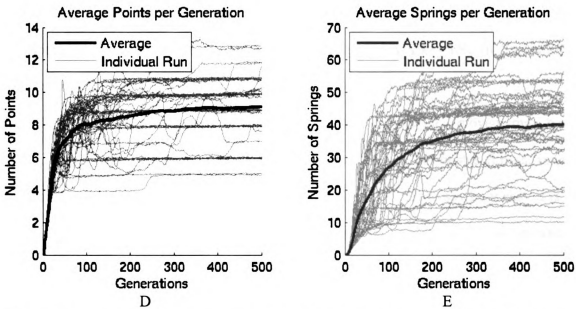


Figure 3.8. Average phenotypic trait change across each population over time for the traits (A) fitness, (B) xz-distance moved, (C) height, (D) number of point-masses, and (E) number of springs.

From the graphs in Figures 3.7 and 3.8, height appears to be the only trait where the average has converged, while the averages of the other traits are still gradually increasing. This supports the argument that, by 500 generations, a given run is less likely to stagnate in WhirlingDervish because it is clearly exploring alternate or improved phenotypic avenues, which corresponds to the continued improvement in fitness. While stagnation does not appear to occur, it is interesting that the averages begin by rapidly increasing but soon settle out to a slower rate of change. This may indicate stagnation for all runs may occur in the future or more runs are stagnating over time.

During the first 150 generations, most many traits seem to experience wild oscillations (Figure 13.8), which persist to a lesser degree throughout the entire run. As these fluctuations become less frequent and the rate of change slows, some strata clearly form. The high rate of change in the early generations indicates that a run should persist for at least 200 generations before a stable solution is likely to be found. It also

corresponds to the initial steep increase in average fitness. I suspect the reason for this correlation is that xz-distance moved does not begin to play a key role in fitness until later in a run.

Furthermore, morphological changes other than xz-distance moved, may have little to no selective pressure to increase once distance moved dominates fitness, unless that change also improves the movement behavior. It is difficult to say how easy it is to change the morphology through mutation when it has little remaining selection pressure. Some evidence of this difficulty is apparent in the formation of strata for height, number of points, and number of springs. Clearly changes do occur and the strata are not completely fixed. Unfortunately, to understand this phenomena will require a correlation between morphological change and change in fitness and distance moved.

3.4.2.2 Population Diversity Based on Various Phenotypic Traits

Now that I have shown the lack of stagnation during the time period observed (due to the continual increase in the average fitness and the average lineage fitness), I will qualify this assertion by applying diversity measures to fitness, xz-distance moved, and height. Figure 3.9 shows the richness measure for the three phenotypic traits. It is apparent that richness for three traits gives the same information. Therefore, fitness is chosen to represent the other phenotypic traits for the contribution of phenotypic diversity towards characterizing stagnation in order to be optimally comparable with other evolutionary systems and diversity studies. Figure 3.10 shows the application of the four diversity measures to fitness, and all four give the same information. From the figure, it is apparent that after an initial jump in diversity, it begins to decrease for the rest of the run. This information supports the assertion that stagnation has not yet occurred.

However, the trend of decreasing phenotypic diversity suggests that stagnation may be likely to occur for longer runs.

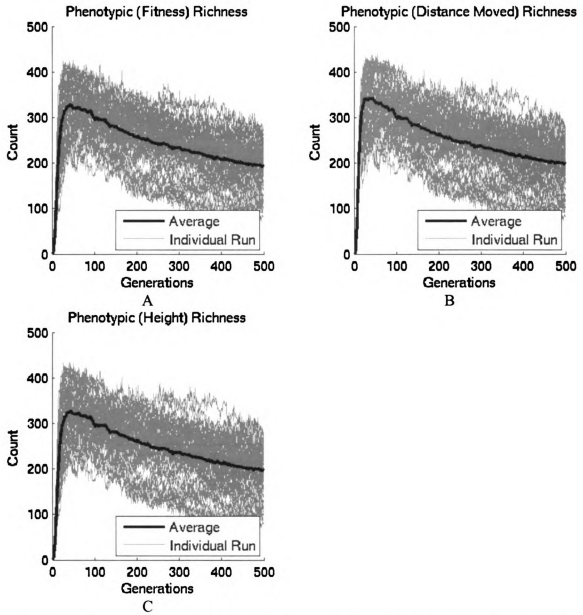


Figure 3.9. Richness measures of (A) fitness, (B) xz-distance moved, and (C) height.

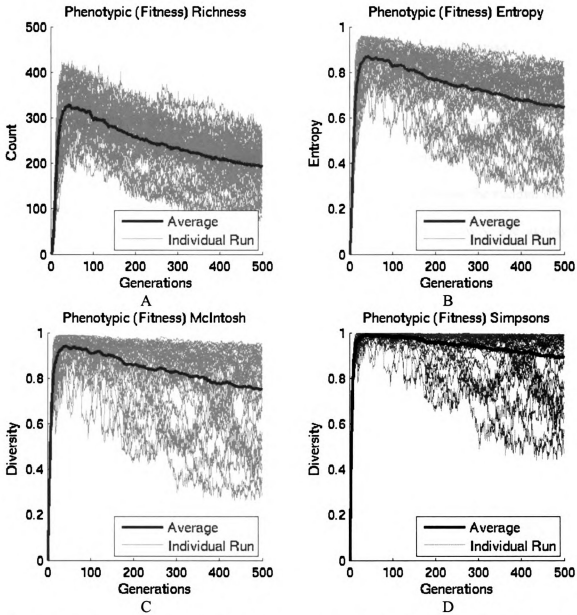


Figure 3.10. The four population diversity measures applied to fitness. (A) richness, (B) entropy, (C) McIntosh, and (D) Simpsons.

3.4.2.3 Average Inferiority and Fitness Distributions

By using inferiority and the distribution of fitness values at discrete time points, I will now examine the variation of phenotypic traits. The inferiority of an individual is the difference between its fitness and the best fitness in its population. Average inferiority is a rough estimate of phenotypic variation and is a strong indicator of when a

population is going through a transition. Thus, as the quantity of improvements decreases or the selection pressure is too light, average inferiority will decrease.

Figure 3.11A shows average inferiority over time for each run and Figure 3.11B shows average fitness over time. While some of the increases in average inferiority are muffled in the noise of the others, a number of sharp increases, many of which are spikes, stand out. Using the outlier, it is clear that a sharp increase accompanies the sudden jump in average fitness. As this average fitness continues to rise more slowly, the average inferiority begins to decrease. Since I know that selection pressure is low and decreases over time, successful individuals should produce more offspring with neutral mutations, thus imparting similar fitness and lowering inferiority.

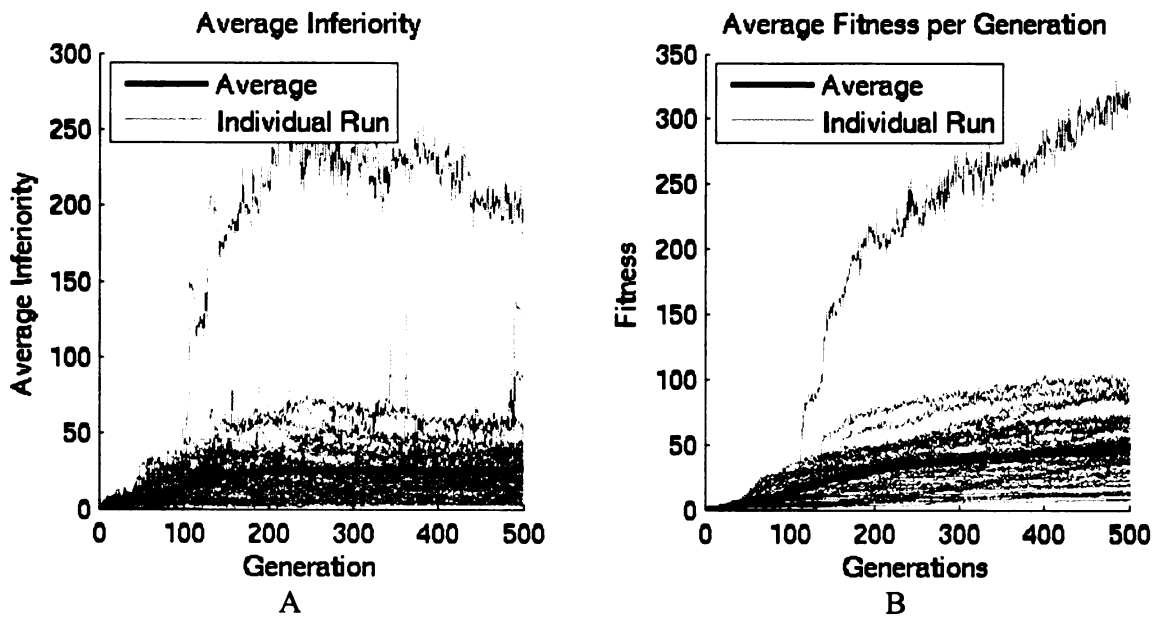


Figure 3.11. (A) Average inferiority per generation, (B) Average fitness per generation

The decrease in average inferiority of the outlier suggests that the run begins to lose phenotypic diversity. Examining the average of the average inferiorities (Figure 3.12) may shed some light on what is happening in the system. For instance, why does the average of the average inferiorities increase and then level off? To answer this

question, seven marks are placed along the average of the average inferiorities at different points in time with three on the increase, three on the level portion, and the last one.

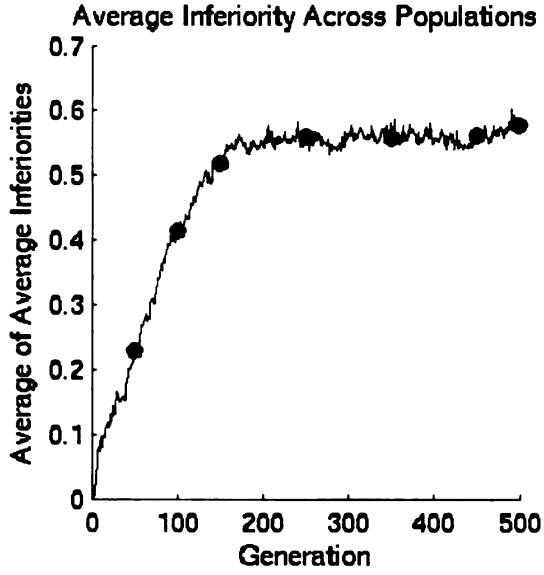
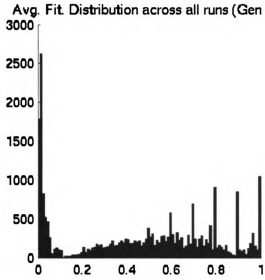
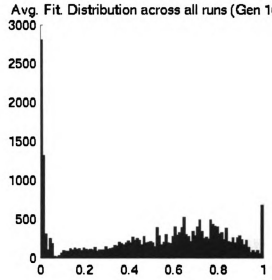


Figure 3.12. Average Inferiorities across populations

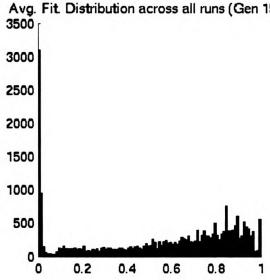
Then, for each of the seven time points, I found the normalized fitness distribution averaged across all fifty runs (Figure 3.13). The idea for using fitness distributions is partially inspired by the work of Popovici and De Jong (2003) who used fitness distributions as a method of characterizing evolutionary algorithms. I use the fitness distributions to help determine the driving force for the change in the average of average inferiorities.


Avg. Fit. Distribution across all runs (Gen 50)
3000
2500
2000
1500
1000
500
0
0
0.2
0.4
0.6
0.8
1

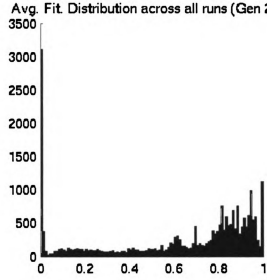
A


Avg. Fit. Distribution across all runs (Gen 100)
3000
2500
2000
1500
1000
500
0
0
0.2
0.4
0.6
0.8
1

B


Avg. Fit. Distribution across all runs (Gen 150)
3500
3000
2500
2000
1500
1000
500
0
0
0.2
0.4
0.6
0.8
1

C


Avg. Fit. Distribution across all runs (Gen 250)
3500
3000
2500
2000
1500
1000
500
0
0
0.2
0.4
0.6
0.8
1

D

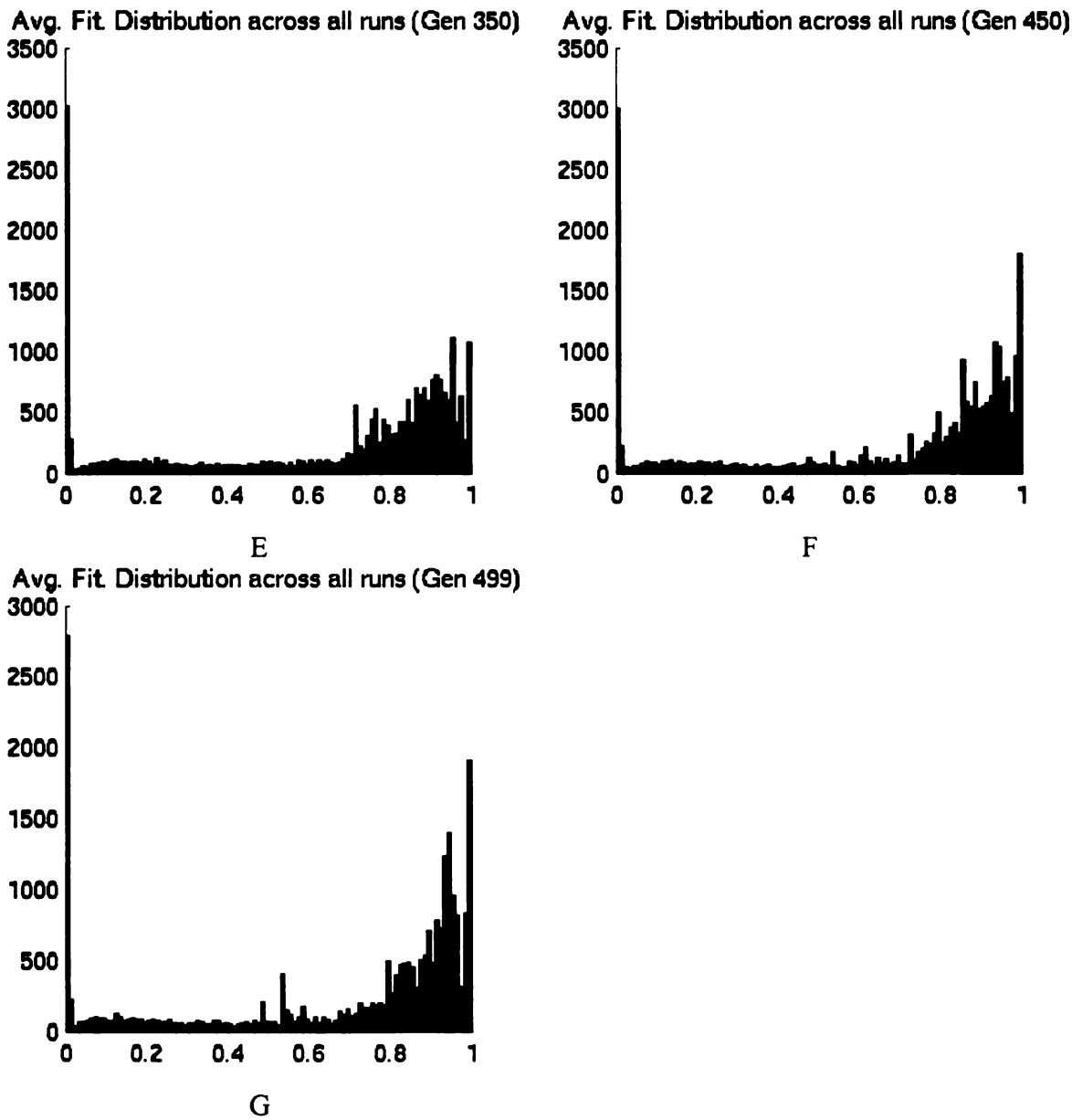


Figure 3.13. Average fitness distributions at generational time points (A) 50, (B) 100, (C) 150, (D) 250, (E) 350, (F) 450, and (G) 499.

The average fitness distributions are constructed from the fitness values of an entire population across all runs for a given point in time. First, each population scales its fitness values by dividing by the maximum fitness within that population. Once scaled, all of the fitness values are lumped into a larger set of fitness values that range from zero to one. This essentially increases the sample size represented by each histogram to the

number of runs times the population size (in this case 50 * 500 = 25,000). One of the biasing factors is the fact that there will always be a minimum number of values at 1.0 equal to the number of runs. However, the relative fraction of data points this represents in terms of the total number of data points is small enough to ignore. Also, the distribution only scales the fitness values, it does not shift them. Thus, any values that fall into bins at or near zero must have those low fitness values.

The change in the average inferiority across populations can be explained by the trend in the fitness distribution over time. Figure 3.13 shows the fitness distributions in WhirlingDervish tend to be bimodal around 0.0 and a positive transient mode. In the early stages of the run, the second mode begins to move toward the other end of the fitness spectrum. Once the mode has established itself near 1.0, its movement is slowed and lesser fit individuals are gradually pulled toward it. Even though the right mode is pulling more and more individuals toward it, the left mode still retains a fair chunk of individuals with low fitness, which I assume correlates with the mutational trends shown in Figure 3.2A. The changing fitness distributions trends explain the changes in the average inferiority across populations. The increase in the average of the average inferiorities corresponds to the second mode shifting to the right, where the curve levels off when both modes are near their final positions. The fluctuations in the average are the ebb and flow of a small portion of the individuals between these modes.

3.5 Conclusions

From the results discussed in Section 3.4, many conclusions can be drawn about the WhirlingDervish system. In particular, this chapter focuses on determining whether or not stagnation and convergence occur. The data clearly demonstrated that no

convergence occurs, at least in the time frame observed, as all of the genetic diversity measures showed high and increasing diversity. After looking at genetic variation, I determined that the low selective pressure was a primary factor in the high diversity. In terms of morphological convergence, it appears that on average individuals continue to change slightly over time. However, the measures were too noisy to make strong assertions about convergence on a fixed morphology, or lack thereof.

The data supporting the likelihood of a given run to stagnate are not as strong as the support for convergence, but still indicates that stagnation is unlikely. It is clear that, on average, the fitness is continuing to increase, and the other phenotypic traits continue to change over time. However, when the system is viewed in terms of variation, it appears to be losing phenotypic variation, indicating movement toward future stagnation. However, my results from testing inferiority and the change in fitness distributions over time, throw doubt on how likely it is for a stagnation event to occur in the future. The other stagnation related results show that the system is in a state of flux in the early generations, and that under the experimental setup given above, a run should always go for at least 200 generations in order to find a stable result.

Other conclusions drawn from this chapter relate to the use of diversity and variation methods and their usefulness. In terms of population diversity, little if any extra information was found using more than one measure for both genetic and phenotypic diversity. The measure that both represented the most information and provided the most contrasting results was Shannon diversity (entropy). Therefore, I will use this measure as the primary measure of population diversity for WhirlingDervish data. However, without other sources of data to compare, this result may be specific to the experiment conducted

here, and may vary with other experimental designs. Thus, the other measures should be examined before discarding in future work.

The use of fitness distributions did provide useful information about the system. However, due to the scaling of the fitness values in order to produce an average distribution, it essentially strips out the quantitative portion of the data. Thus it can never show an infinitely increasing distance between the two modes, only that the distributions are near maximized. Using the average inferiority appears to compensate for this deficit due to the fact that one mode does not move, thus increasing average inferiority means the mode is either moving to higher fitness values or pulling more individuals from the other mode. However, the average inferiority cannot characterize a fitness distribution. Therefore, it seems that these measures are more informative together than they are individually, at least in terms of stagnation.

In conclusion, this chapter has shown that convergence did not occur in WhirlingDerivsh, and as time continues, the selective pressures continue to decrease. It has also shown that at least under the design of this experiment, stagnation is unlikely. Lastly, these measures have shown themselves to be potential tools for characterizing convergence and stagnation for the WhirlingDervish system.

Chapter 4 Understanding the Fitness Function

Drawing upon my previous work with diversity and variation measures (Chapter 3), I analyze the default fitness function in WhirlingDervish. I systematically focus on each fitness component to understand its role in the evolutionary dynamics of the population. My overarching goal is to find a generalized computational method of comparing systems that evolve 3D structures and behaviors.

Few studies have been conducted on fitness schemes in systems that evolve 3D morphology and behavior. The primary study is by Karl Sims (1994b). He published two separate works, one on explicit fitness functions to evolve nontrivial individuals capable of walking, swimming, and jumping, and one on the use of competition to evolve behaviors centered on gaining possession of a cube placed in the environment. In the competition experiment, Sims examined different types of tournament competition and the use of species, or islands, on the evolved creature behavior. Through competition he expected to see an increase in behavioral complexity from the creatures evolved versus the ones evolved under an explicit fitness function, but did not explicitly test this hypothesis. The evolved behavior may have been sufficiently different that a good comparison was not possible. Regardless, Sims did demonstrate that both fitness methods could successfully evolve interesting behavior.

Since Sims' work, most 3D morphology and behavior systems use an explicit fitness function based primarily on the distance an individual moves. Other phenotypic traits are also used but tend to vary from system to system. The premise for most fitness functions is often derived from Sims's (1994a) work, which used velocity as the fitness function for both swimming and walking. In at least one of Hornby and Pollack's

(2001b) systems, the fitness function took the difference between the xz-distance moved and the total distance all of the physical components were dragged along the ground, which was intended to encourage a rolling or stepping behavior. The fitness function used in the Framsticks system (Komosinski, 2000) is based on a few phenotypic characteristics such as age, velocity (distance over age), and distance traveled by the center of mass. In the recreation of Sims's work by Taylor and Massey (2001), they used multiple fitness functions based mostly on the distance an individual moved. Due to the wide usage of a static fitness function based on distance moved, I designed a similar fitness function for the WhirlingDervish system.

The default fitness function in the WhirlingDervish system is an explicit fitness function, which I will call the "Base" fitness function. The goal of the Base fitness function is to evolve a body structure capable of non-trivial, mobile behavior. Due to the abstract nature of this goal and a current inability to automatically categorize mobile behavior and structure, I designed a function based on a set of phenotypic characteristics that each reflect an aspect of the evolutionary goal. Some phenotypic characteristics are also used as constraints and fitness penalties to avoid specific trivial solutions and reduce the computational time required to test the population each generation.

The reasoning behind the Base fitness function is that different phenotypic aspects are important at different generational times. After initial testing, the sole use of the xz-distance distance traveled turned out to be insufficient due to evolutionary domination of individuals that started out as compressed springs that threw themselves and then stopped moving. With the idea that consistent movement over time requires the maintenance of three-dimensional structure, I added four phenotypic traits to the fitness function. These

traits are the number of point-masses, the structure's overall height, the height of its center of mass, and the requirement that the final overall height must be at least one third of the original overall height. The first three of these traits are limited in their fitness contribution because structure is not the primary focus. The fitness caps are a maximum of ten point-masses and a maximum of nine meters for both the overall and center of mass height.

Some additional constraints are designed to prevent wire-frame individuals from using extreme physical properties to their advantage, such as bounds on spring constants and the actual mass of a point-mass. Other constraints are meant to keep run times consistent and manageable, and the body structure within reasonable bounds. The genetic language handles some of these constraints by not allowing individuals to evolve certain parameters away from the default value. The other constraints are treated as penalties in the fitness function.

The three fitness penalties used in the Base fitness function are a bounding volume, a maximum y-value, and a cap on the number of springs in a phenotype. The penalty for the number of springs uses a cap set at fifty-four (one less than a complete graph for eleven point-masses). For each spring above the cap, a penalty of one is deducted from the final fitness value. However, this only discourages wire-frame individuals with more springs, it does not prevent such a situation given an otherwise sufficiently good solution. For the bounding volume of a given body, a maximum value was set of 26.0m for the length of the diagonal of the box, which equates to a bounding box with dimensions of 15m x 15m x 15m. The maximum y-value is set to 30m and is used to prevent individuals from throwing themselves so far into the air that they don't

land during the evaluation period.

Figure 4.1 shows a pseudocode snippit that calculates the Base fitness function including both the positive and negative factors.

```
fitness = 0.0;
if (final_height > 0.3 * original_height)
        fitness += 3.0 * xz-distance
fitness += 0.0254 * number_of_points
fitness += final_height
fitness += 3.0 * center_of_mass_height
if (number_of_springs > 54)
        fitness += 54 – number_of_springs
if (bounding_box_diagonal > 26.0m  ||
   max_y_coordinate_value > 30.0m ||
  fitness < 0.0)
        fitness = 0.0
```

Figure 4.1. The default "Base" fitness function

4.1 Experimental Design

This experiment examines the impact of the different factors that comprise fitness, and how well diversity and variation measures can provide that differentiation. To determine the impact of these factors, I take the Base fitness function (Figure 4.1) and break it up into six variants, which I will call run Sets I-VI. The first variant, Set I, was created so that I could examine a fitness function that uses only the distance moved fitness component, the spring penalty, and other constraints. The purpose of this variant is to test the raw evolutionary power of this metric, and is of particular interest due to the fact that a similar measure is used in most related evolutionary systems.

The other variants, II-VI, examine the impact of each factor independently by knocking out that factor from the Base fitness function. Table 4.1 displays these knockouts.

Table 4.1. A chart of the fitness components used in the fitness function for each set of runs. The left hand column corresponds to the fitness factor labels given in Figure 4.1, where the Base fitness function uses all of the factors.

Component							
(a)	Base			III	IV	V	VI
(b)	Base	I	II	III	IV	V	VI
(c)	Base		II		IV	V	VI
(d)	Base		II	III		V	VI
(e)	Base		II	III	IV		VI
(f)	Base	I	II	III	IV	V	

Each set consists of fifty replicate runs that use identical configurations other than the fitness function. The configurations use a population size of 500 individuals that are simulated for two minutes to compute fitness. With the initial population consisting entirely of empty genomes, the run continues for another 499 generations. An entirely new population is created each generation using fitness proportional selection with an elitism of one. Once an individual is selected to generate an offspring, its genome is copied instruction by instruction into the genome of the offspring. This process occurs external to the individual; the individuals do not technically self-replicate. As each instruction is copied into the offspring's genome, there is a chance mutation will occur based on a rate computed prior to the copy process. That rate is genome length dependent and set such that there is an average of one mutation per offspring. The four types of possible mutations are point, insertion, duplication, and deletion, but no crossover is used.

4.2. Results and Discussion

When comparing the results of each run set, fitness is not an appropriate choice because each function has slightly different range. Instead, I select a common phenotypic

trait that is important to the desired output from the system. In this situation, I choose to use the average distance moved across all populations for a single set of runs. I then use the average to compare all set of runs as shown in Figure 4.2.

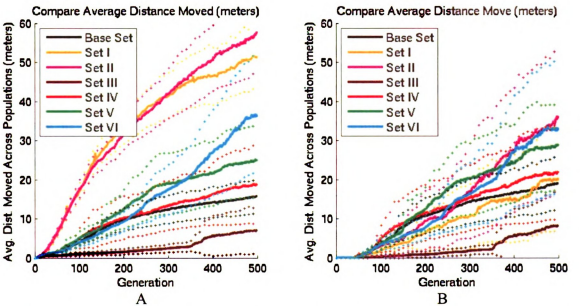


Figure 4.2. Average distance moved vs. time averaged over the 50 populations in each run set. The pluses are the 95% confidence interval for the respective run that shares its color. (A) The average distance moved over the entire two-minute evaluation period, and (B) The average distance moved in the last minute of evaluation.

It is evident from Figure 4.2A that three variants are significantly different from the Base fitness function, with Set VI, no spring restriction, only establishing itself from the Base fitness function near the end of the evolutionary process. Runs I and II, the variant with just distance moved and the variant with no height restriction for gaining fitness for moving, show the ability to evolve individuals that can move much farther on average. The possible reasons are that requiring height restricts some of the intermediate forms that would allow an individual to move further or faster. Admittedly, height is an additional selective pressure and trying to evolve more than one factor at a time can slow progress. Evolving under movement only, it is possible that the population has access to

evolutionary routes to better movers early in the evolutionary process.

While these alternative fitness functions seem to show improvement, they encourage individuals to move further faster with no concern for maintaining structure. The purpose of requiring structure at the beginning and end is to encourage consistent movement for long periods of time. Figure 4.2B shows that while both these sets of runs are able to evolve consistent movers, it is apparent when comparing them in terms of moving in the second half of the evaluation, the resulting differences with the Base fitness function disappear. The early movers in Sets I and II are more successful on the whole because they fling themselves at the beginning of the simulation due to their initial structural configuration, but then lose much of their structural integrity. Historically, their early individuals were likely to simply throw themselves and stop moving. Generations later, their progeny were then likely evolve the ability to roll or repeat the jump, but the historical jumping component is likely to be maintained evolutionarily.

The only set of runs that showed significant difference under both methods of comparing distance moved was Set III. This set uses a fitness function that is only missing the reward for point-masses. Under such a fitness function, the individuals have no selective pressure until they are able to move or have height, which can cause a long lag-phase before the evolution of movement can begin.

Next, I compare the different sets of runs using diversity and variation methods. There appears to be little difference between the runs in terms of genetic and phenotypic diversity (Figure 4.3). The only exception is Set III, which has a significant lag from the others in terms of phenotypic diversity (Figure 4.3B). It is interesting that there is little diversity change over time.

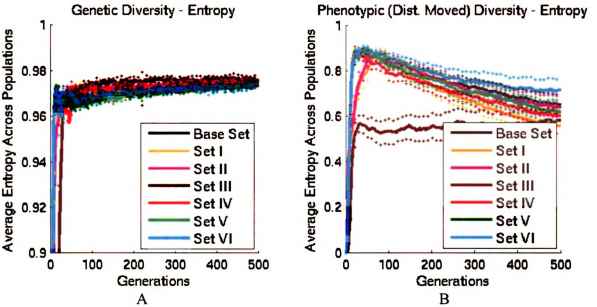


Figure 4.3. Diversity of (A) genotypes and (B) the distance moved phenotypic trait, for each set of runs. The pluses are the 95% confidence intervals for their respective run associated by color.

Figure 4.4 shows a qualitatively similar result for the dissimilarity metric. While the other sets of runs are increasing in average dissimilarity, individuals in Set III are continually becoming more similar. This may be showing the randomness at the beginning of a run and the accumulation of useful instructions. Only toward the end of the run, is the amount of similarity is approaching that of the other sets of runs.

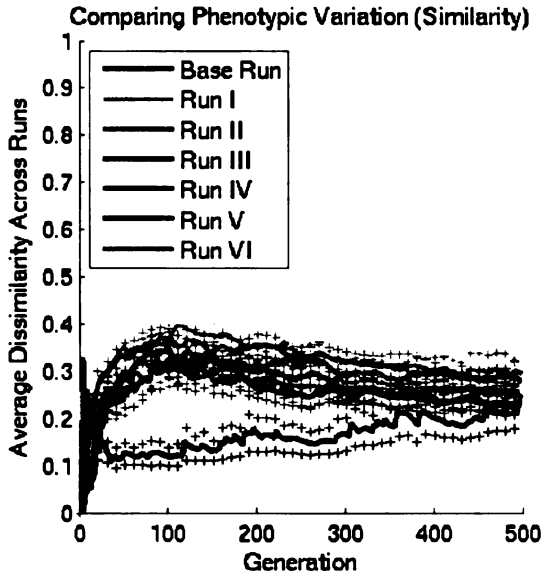
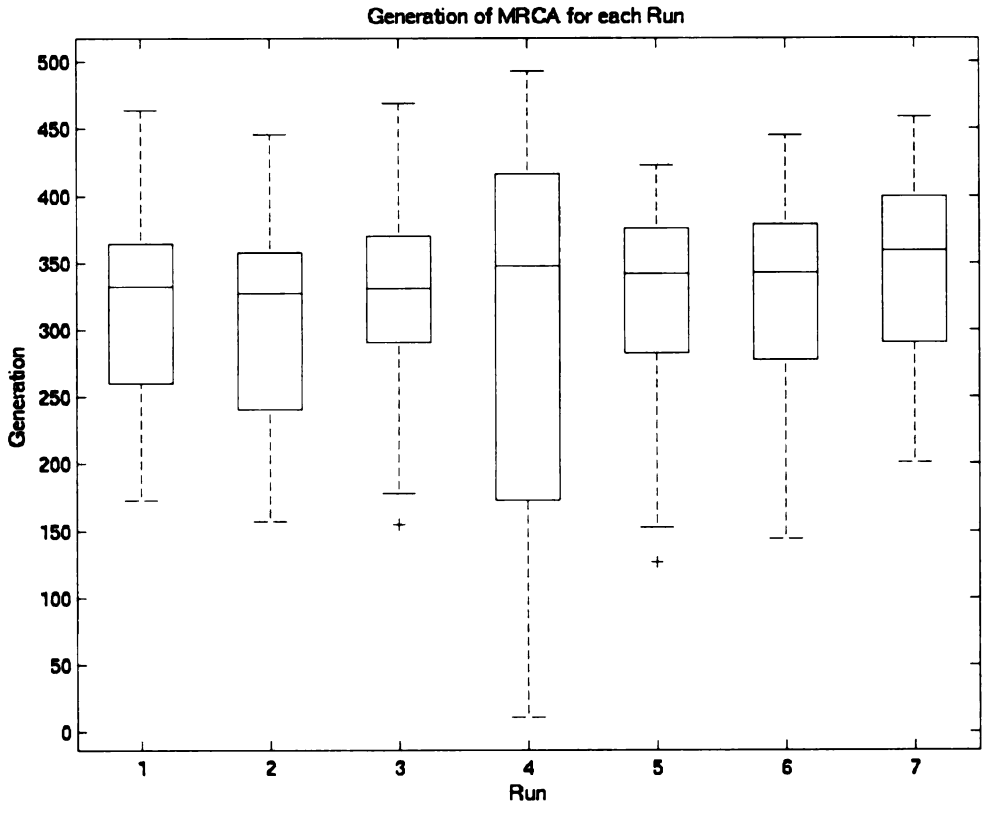
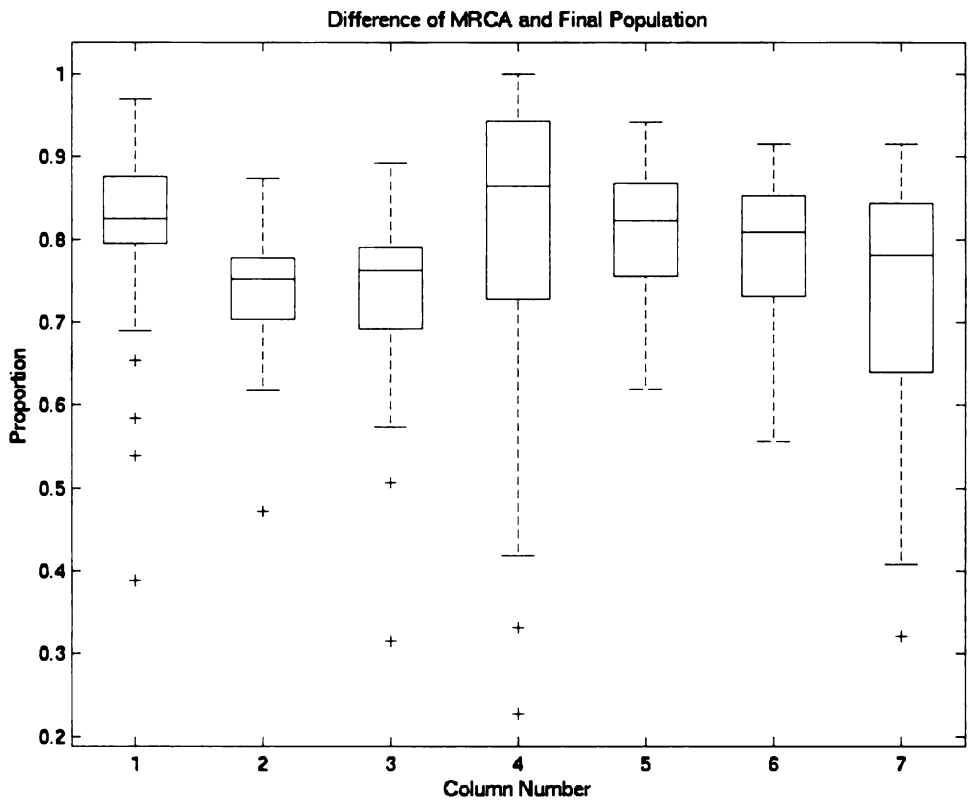


Figure 4.4. The average dissimilarity across all populations for each Set. The pluses represent the 95% confidence interval for their respective runs according to color.

The other genetic variation method I used to compare the fitness function variants is the average MRCA generation and the average dissimilarity between the MRCA and the final population (Figure 4.5). Both graphs show no significant difference between the different sets of runs. Once again, Run III varies slightly from the others with much more variable values. In conjunction with the results from above, I suspect that all of the information about Run III is pointing to such a small level of selective pressure that it is almost a random process, at least for the majority of the evolutionary process.


Generation of MRCA for each Run
Generation
Run
0
50
100
150
200
250
300
350
400
450
500
1
2
3
4
5
6
7

A



B

Figure 4.5. Phenotypic variation (based on distance moved) using MRCA measures. (A) The generation of the MRCA for all fifty runs for each set, and (B) The average dissimilarity between the MRCA and each individual in the final population for all fifty runs within the set.

4.3. Conclusions

In this chapter I examined the various factors that comprise the default fitness function. I clearly showed that knocking out any single factor from the Base fitness function had little differentiable impact on the average distance moved. The exception was the removal of the fitness component associated with the number of point-masses, which had a significant negative impact on the ability to evolve long distance movers. On the other hand, removing the height condition on the ability to gain fitness for moving had a significant positive impact on the ability to move. Using only the fitness

component for distance moved also had the same impact as removing the height restriction for gaining fitness for moving.

While it is interesting that the simplest fitness function, only distance moved, was one of the best for evolving movement, those improvements were lost when comparing the impact of the various factors at evolving the ability to also move during the second half of the evaluation period. This suggests that some of the movement behaviors are not a desired solution type, typically non-consistent movers. However, it is impossible to tell what specific behavior styles evolved that gave these results. Thus with the exception of the removal of the reward for point-masses from the Base fitness function, there appears to be no significant impact for any given factor or even over using only distance moved, at least under the state of WhirlingDervish used for this experiment.

The other purpose of this chapter was to discover if diversity and variation measures, by characterizing convergence and stagnation, would provide a method to compare the evolutionary results. With the exception of run III, all of the runs had basically given the same measurement results. There are three possible situations at this juncture. The first possibility is that these measures are not useful for this task. The second is that the fitness factors have little, if no, differentiable impact on the evolutionary results, given the state evolutionary system used for this experiment. The data provided here can not differentiate which of the two possibilities are more likely. The third possibility is the low selective pressure, observed in Chapter 3, may be obscuring useful information.

Chapter 5 Structure and Behavior Case Studies

I choose six runs out of the fifty that used the Base fitness function as case studies to perform a more detailed examination of evolved structure and behavior. Using these runs, I focus on three areas in this chapter. First, I provide a characterization of structure and behavior for select individuals in each run. Using these characterizations, I then examine both the inter-population variation and the historic variation of structure and behavior. Finally, I examine their genetic diversity and how it relates to the structures and behaviors that evolved. These topics are important because they reveal more details about the evolution of structure and behavior, which is the true goal of WhirlingDervish.

The primary data for this chapter are the manual characterizations of various individuals. For each characterization I interpret an individual's structure and behavior by viewing it. There are two major problems with using this method: the many thousands of individuals in a given run and the difficulty in providing consistent analysis of structure and behavior. When dealing with the sheer quantity of individuals, I attempt to use available information to help me reduce that set to a manageable size. The primary forms of information at my disposal are phylogenetic trees, including the full details for the lineages of each individual in the final population, and historical fitness data. The phylogenetic trees allow me to limit myself to samples from each group of closely related individuals within the final population. Also, the lineage and fitness data allow a similar sampling of ancestral individuals for comparison. I then take these individuals and compare them on both structural and behavioral levels. While this technique is clearly subjective, I take great pains to make it as consistent and formalized as possible. While

more automated analysis methods would be preferred, they often have severe flaws and require significant tailoring to the problem (and such fine-tuning would make for a good thesis on its own).

The problem with consistently analyzing individuals is that the evolved structures are dense, it is difficult to characterize their components, and it is nearly impossible to determine how those components contribute to the perceived behavior. Any given point-mass can have many springs, which, in turn, leads to a large quantity of crisscrossing lines when an individual is viewed from an outside perspective. The crisscrossing of lines can also make it difficult to observe point-masses in the interior of the body. Besides the problems of viewing key pieces of a structure, another problem is the potential non-linear movement of an individual, which can have a confounding effect by transitioning from one type of behavior to another part way through its lifetime. This can occur in different situations such as periodicity differences in spring fluctuations where some point-masses will fall out of synch with one another causing an overall change to the behavior.

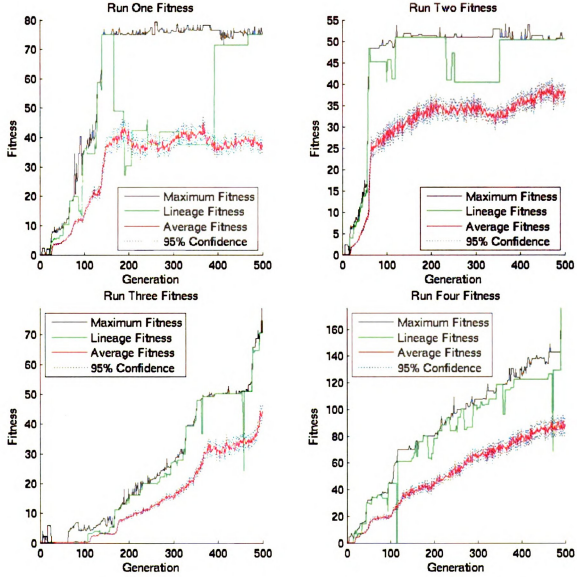
To address the analysis problem, I consistently apply a specific method of analysis to an individual. I first identify key point-masses or groups of point-masses. Next, I divide up the overall behavior into a series of smaller more specific behaviors that are executed in a deterministic order. Once each sub-behavior has been identified, I associate it with the key point-masses identified earlier to determine a rough idea of how that behavior works. The overall generic behavior is characterized in terms of the periodic execution of the sub-behaviors and utilization of specific point-masses. Finally,

to avoid issues such as transitioning between different overall behaviors, I reserve characterization until the first set of consistently repeating behaviors is apparent.

The rest of this chapter involves the examination of behavior for individuals in the final population and certain historic individuals. Using lineage and fitness data, I perform an analysis of structural and behavioral variation. I also examine genetic diversity at informative positions in the genomes, which involves the knockout of non-functional instructions from the genome. These knockouts help me gauge the use of some genetic diversity measures used in Chapter 3 for measuring variation of structure and behavior over time, but especially in the final population.

5.1 Case Study Setup

I chose six runs from the fifty used in Chapter 3 (under the Base fitness function). These six runs were selected in three pairs: two runs that appear to have stagnated (row one), two runs with continually increasing fitness over time (row two), and the two runs that achieved the highest fitness observed in the final generation (row three). I designated the first pair of runs as stagnated due the overall lack of change in lineage and maximum fitness, and the small, noisy increase in average fitness. The purpose to this selection method using fitness (Figure 5.1), rather than random sampling, is to develop and intuition as to if there is a correlation between the lack of variation in structure and behavior and the stagnation of a run. For example, it would be understandable if the stagnated runs have low structure and behavior variation in the final population.


Run One Fitness
Fitness
Generation
Maximum Fitness
Lineage Fitness
Average Fitness
95% Confidence
Run Two Fitness
Fitness
Generation
Maximum Fitness
Lineage Fitness
Average Fitness
95% Confidence
Run Three Fitness
Fitness
Generation
Maximum Fitness
Lineage Fitness
Average Fitness
95% Confidence
Run Four Fitness
Fitness
Generation
Maximum Fitness
Lineage Fitness
Average Fitness
95% Confidence

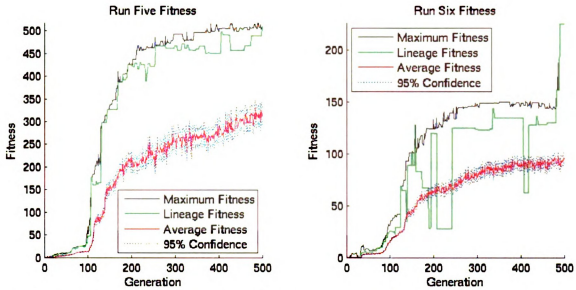


Figure 5.1. Fitness vs. time for three fitness metrics across six independent runs. The fitness metrics used are maximum fitness at each time point, the fitness of the individual along the lineage at each time point, and the average fitness at each time point. The graphs are paired up in the following order: stagnated (runs one and two), continues to evolve (runs three and four), and achieves maximal final fitness observed (runs five and six).

In order to make this study consistent across identical individuals, I recompute the fitness of each genome to remove the stochastic perturbations that are normally applied to each point-mass before and individual's evaluation. This means that exactly identical phenotypes will always produce exactly identical measurements – something that the small perturbations would normally cause problems with.

5.2 Results and Discussion

Characterization of evolved individuals is the first step in examining the evolution of structure and behavior. I begin by characterizing the most-fit individual from each run. These characterizations form a reference point in which to describe other individuals within their respective populations. To understand these characterizations, it is important to keep in mind the importance of rhythm for locomotion. Rhythm is the interaction of two or more components that move in a periodic fashion. Thus, without reactive control

over its body, an individual is forced to rely on very regular cyclic behaviors in order to repeatedly execute their locomotive behavior.

5.2.1 Description of the Most-Fit Individual in Each Run

The most-fit individual in run one (Figure 5.2) forms a complete graph with three point-masses comprising the top and six on the bottom. The structure appears to be approximately rectangular and is narrow in the direction of movement. The three top point-masses cross the narrow width of the body near one end of the rectangle, and the infrequent, slight lifting of the bottom point-masses at the other end cause the body to rotate. The rotation leads to a non-linear direction of movement that is similar to an overall elliptical movement. The movement is driven by the lead point-mass on the top which points in the forward direction of movement. The other two point-masses are off center from the direction of movement and move in large circular motions. The large circular motion means that the point-mass will provide more lift to the point-masses on the bottom nearest to it and farther from the others. As a point-mass on the top picks up those on the bottom to varying extents, the bottom ones are moved to new locations. Due to the leading point-mass, this picking up and moving of the bottom ones is what gives the individual its ability to move in a forward direction.

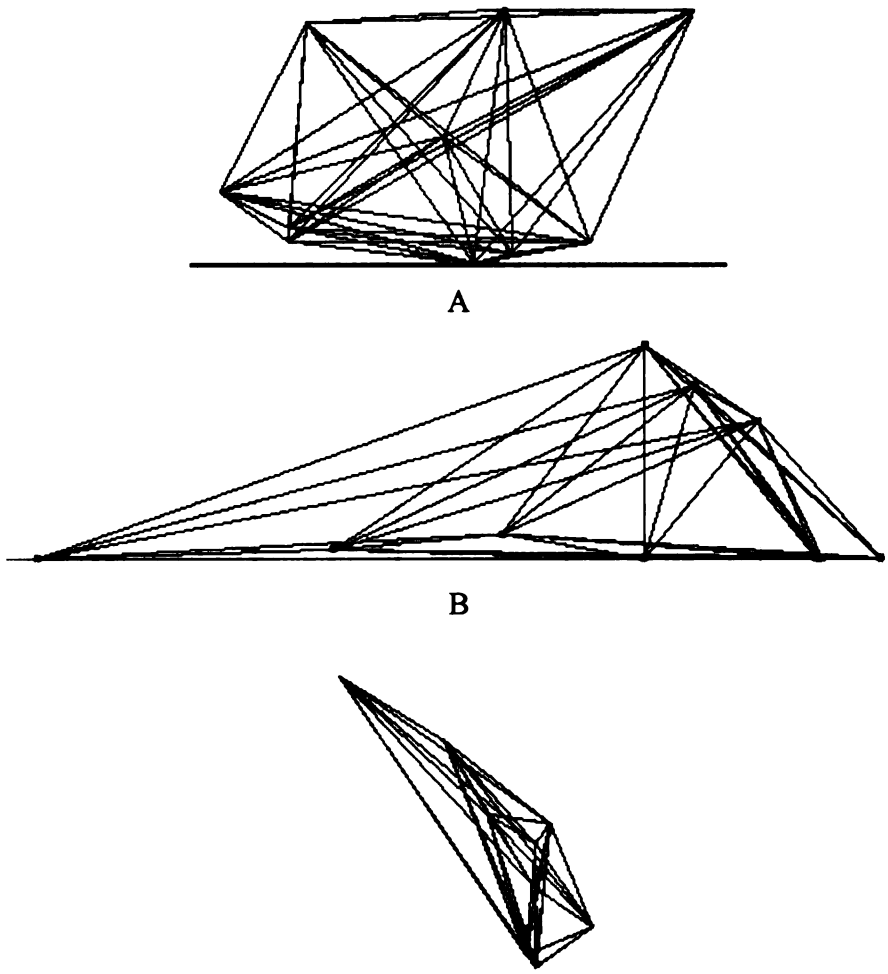


Figure 5.2. (A) A side view perpendicular to the direction of motion of the most-fit individual from run one. The direction of movement is to the left. The individual is raising and thrusting its front forward, and will soon rotate down, pulling the back end forward. While the front is rising, the back end is moving towards the ground. (B) A view parallel to the direction of motion. (C) A top down view of the individual. There are slight size discrepancies because each image had to be taken a different point in time and during different points in the rhythm.

The most-fit individual from case study two (Figure 5.3) forms a lopsided pyramid consisting of a complete graph with nine point-masses. The behavior relies on three components: the lead points, the backbone, and the base. The two point-masses in the lead, point in the direction of motion. They are then pulled up and thrust forward.

Pulling up of the lead point-masses is accomplished by the expansion of the backbone while pushing the base down onto the ground and leaning back. Once the leading point-masses have been thrust forward, the backbone rocks forward as if rolling on the base. Two point-masses from the base are then picked up into the air and two remain on the ground, which, in conjunction with the leading two, pull the body forward through contraction. The cycle is then repeated, giving approximately linear forward movement. It also appears that given enough time, it will rotate approximately 135 degrees to its right, and once again proceed in a linear direction. However, the evaluation period during evolution is only two minutes, thus the rotation would not be selected against. Figure 5.3 shows this individual in the process of lifting the leading point-masses before thrusting them forward.

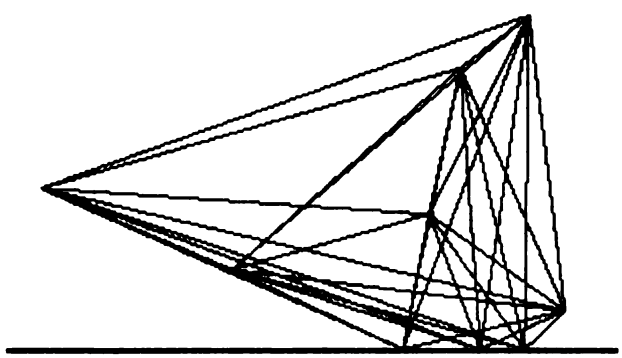


Figure 5.3. A side view perpendicular to the direction of movement of the most-fit individual from run two. The direction of movement is to the left. The slightly curved shape of the base is visible at the bottom. The curve allows the individual to rock back, thrusting the leading component forward, then rock forward, dragging the back end forward through contraction.

A similar behavior is observed in the most-fit individual from run three (Figure 5.4). This individual appears to also be a connected graph, but consisting of approximately twelve point-masses. The primary difference between the two individuals

is run two utilized a backbone type structure where this individual utilizes a pentagram type structure. The pentagram structure rolls back and forth along the direction of motion, lifting and thrusting the leading point-masses out ahead of the body. The other point-masses are connected to the pentagram opposite the leading ones but off the ground, which contributes to the ability of the pentagram to rock. With the few point-masses in contact with the ground, the distance that the leading point-masses are thrust is farther than the contracting distance of the point-masses, giving the overall forward motion of this individual.

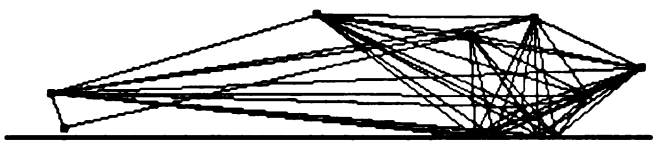


Figure 5.4. A side view perpendicular to the direction of movement of the most-fit individual from run three. The direction of movement is to the left. You can see the interesting core component that consists of many pyramids pointing in different directions. It is the interactions of these pyramids that allow the rocking of the central component for lifting and thrusting.

For run four, the most-fit individual (Figure 5.5) is able to lift its entire body into the air, essentially, throwing itself forward. This individual has the same general configuration as the individuals from run two and three, from the top down it appears to be a triangle or diamond pointing in the direction of motion. What makes this individual unique is that it has two important sets of point-masses that alternate positions in the air. The first set forms a type of backbone, which rotates down to lay almost completely on the ground. As it rotates downward, a point-mass in the front pops up into the air. At some instance during movement, this forward point-mass is the only one in the air, and

provides the leverage used by the rest of the point-masses on the ground to pull themselves up and forward during contraction. The movement exhibited by this individual is fast, but unstable. Given enough time, it collapses onto the xz-plane unable to recover. I suspect this occurs if the body gets out of synch enough that it cannot raise the point toward the front. However, the time frame for collapse is well beyond the two-minute evaluation period, preventing selection against this instability. Figure 5.5 shows both ends of the body contracting, raising up off the ground.

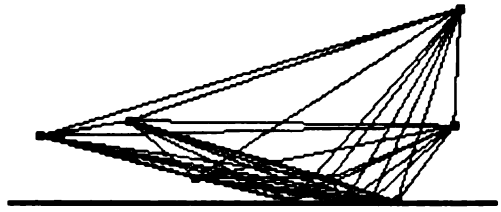


Figure 5.5. A side view perpendicular to the direction of movement for the most-fit individual from run four. The direction of movement is to the left. This individual has less infrastructure in the back end for lifting the front component. Thus, the highest vertical point-mass near the front is key to lifting the other leading point-masses.

In run five, the most-fit individual (Figure 5.6) utilizes a fully connected graph where one point-mass resides far out in front of the rest. The bulk of the point-masses are close together in a large clump. The behavior follows an alternation between first lifting the point in front and thrusting it forward with most of the back point-masses on the ground. Then the forward point mass lands and the majority of the back point-masses are lifted into the air. During the second period, considerably less friction is present allowing the contraction of the back point-masses to provide further movement towards the front. During this contraction phase, the back point-masses appear to almost dance by

constantly switching who is touching the ground, which gives each point more time in the air than touching the ground.

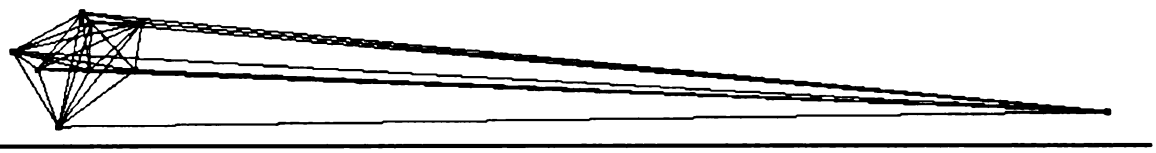


Figure 5.6. A side view perpendicular to the direction of movement for the most-fit individual from run five. The direction of movement is to the right.

The most-fit individual in run six (Figure 5.7) is similar to the one in run five. The primary difference is the addition of one component. This component adds a point-mass connected to the bulk at the back, but sticks out behind. As the front point-masses, there are two, are thrust forward, this point in the back is moved ahead of the bulk. When the bulk is ready to contract with the front ones, this extra point acts a leg, allowing the bulk to be thrust up into the air. This gives an overall affect of lifting the entire back end up into the air, and sometime the whole body. This motion is repeatable, but can get slightly out of synch causing a slow down in movement speed and a rotation 90 degrees to the right. Figure 5.7 shows the individual right before it would rock back, lifting the front point-masses into the air. It also shows the opposite where the leading point-masses have been thrust forward and the body is rocking back to a rest position.

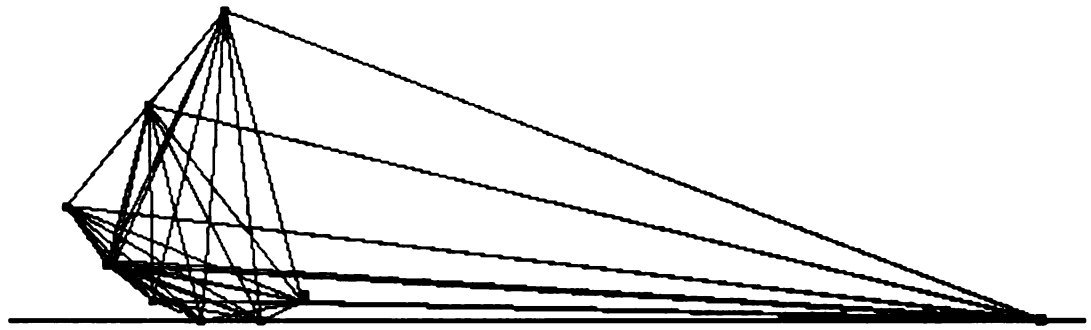


Figure 5.7. A side view perpendicular to the direction of movement for the most-fit individual from run six. The direction of movement is to the right.

One aspect shared by all of the most-fit individuals from each run is a body formation where almost every point-mass is connected to every other point-mass. There are occasional point-masses hanging connected through a single spring to the rest of the body. Also, the use of alternating point-masses in the base components being lifted into the air at different times appears fairly common. Another aspect is that most of the individuals seemed to make use of internal structures such as the pentagram from run three, and levers in run six. All of the structures and behaviors viewed so far have been complex and have shown the importance synchronous cycles play in the dynamics of these individuals.

5.2.2 Variation Within the Final Population

Knowing the structure and behavior of the most-fit individuals within a run is useful, but how much variation exists within a population? To answer this question, I sampled individuals from the final populations guided by the phylogenetic tree for each run. The purpose of using a phylogenetic tree is that it provides a rough clustering of individuals who are closely related. Individuals near each other in the phylogeny are much more likely to have similar, if not identical, phenotypes. The designation of a

cluster is the set of individuals whose common ancestor lived within the previous twenty-five generations. To reduce the time expense required to analyze every individual, I sampled individuals from each cluster consisting of the highest fit individual, an individual with about half the highest fitness, and sometimes a few others with different fitness values. Figure 5.8 illustrates a partial phylogenetic tree containing two clusters.

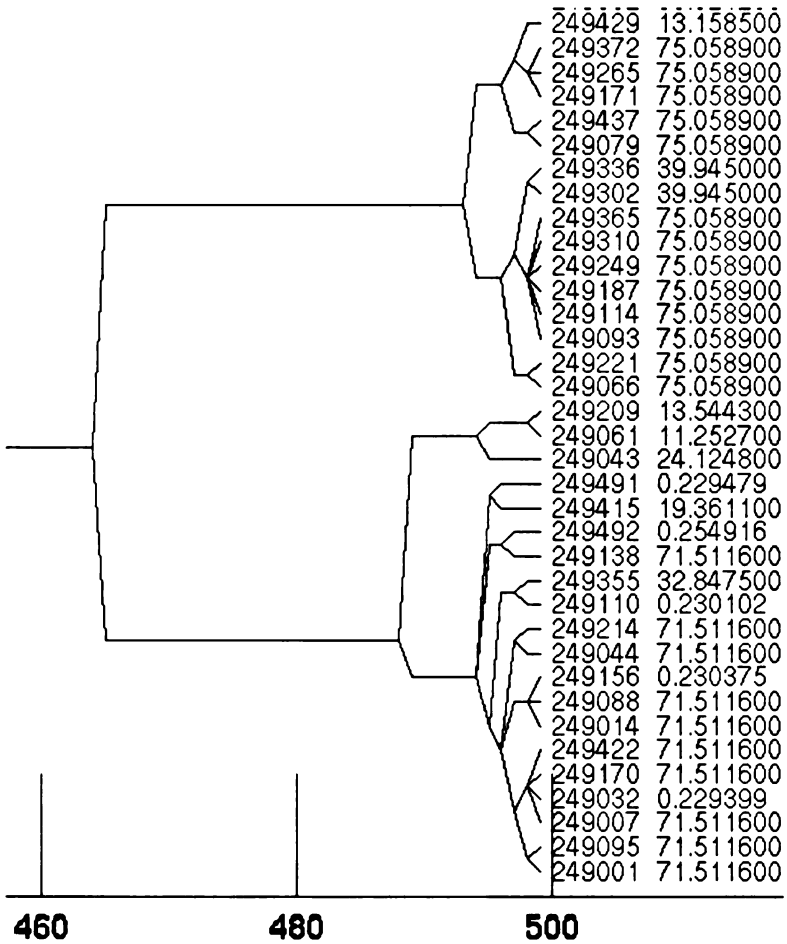


Figure 5.8. A partial phylogenetic tree from run one showing two clusters. The numbers along the bottom represent the generation. The first vertical column is the individual's id and the second column is the fitness of each individual.

In the end, I sampled between fifty and seventy-six individuals from each of the six final populations. The one exception was run two where I only selected eleven

individuals. The reason for this exception was the sheer quantity of individuals that had exactly the same fitness. Each of these individuals seems to have divergent genomes due primarily to neutral mutation, but have identical phenotypes. There is no guarantee that individuals with the same fitness will have the same phenotype. However, fitness is a high precision real value, which makes it very unlikely for multiple phenotypes with the same fitness to be different, especially between individuals in the same cluster. Thus, I took multiple samples of individuals with the same fitness value from different clusters, and included other individuals that had reasonable fitness values.

The examination of all of the sampled individuals from the final populations showed that in five of the six runs there was only one type of structure and behavior present in the final population. The only exception was run six, which showed two distinct types of structure and behavior. Interestingly, the most-fit individual in run six was not the most common phenotype. The most common phenotype was actually more similar to the most-fit individuals from runs two and three, which use the lift of the leading point-masses and then rock forward type of behavior in conjunction with a raised back end.

The variance in fitness is caused most often by changes in the quality and efficiency of executing a behavior. These changes occurred in many different ways such as shifting one point-mass relative to the others or shifting a point-masses position causing a change in rhythm. Changes also occurred through the slight alteration of structure such as the addition of a new point-mass or the removal of a spring. These alterations often led to changes in rhythm, but also changed how the structure was

maintained over time. The alterations also affected behavior, sometimes adding or removing complexity from the phenotype.

With the final populations having minimal variation in general structure and behavior, I wanted to know how long those general structures and behaviors have persisted. To determine this, I compared the most recent common ancestor (MRCA) to the individuals in the final population. With the exception of run six (where a new, more fit phenotype is arising), the MRCA had the same general phenotype as the most-fit individual. For run six, the MRCA had the same phenotype as the most common phenotype in the final population, but not the same as the most-fit phenotype in the population. This implies that the most-fit phenotype in run six evolved after the most common phenotype swept the population. After comparing the two phenotypes in run six, it is clear that the most-fit phenotype evolved by building off and modifying its ancestor. All of the other runs simply contained a sweep, and judging by the fitness increase in the majority of the runs (Figure 5.1), they have only improved on that phenotype.

The MRCAs for the runs occur at different times (Table 5.1) and do not provide enough information about the historical context of these sweeps. Thus, I utilize the lineage fitness for each run to select ancestors for comparison. These comparisons will shed light on evolutionary progression of the structure and behavior found in the final population. I start by picking the most-fit individual from the final population and trace its ancestry back to the first generation. Then, starting at the earliest ancestor in the first generation, I proceed forward along the lineage, finding those individuals who meet certain criteria. The criteria uses the fitness of the final individual to calculate a

percentage of that fitness starting at 5% and increasing by 5% up to 100% or the MRCA, which ever is first. Those whose fitness stagnates early will sample all twenty individuals long before the generation of the MRCA, while the others will only sample about ten individuals.

Table 5.1. Generation of the most recent common ancestor.

Run	MRCA Generation
One	208
Two	356
Three	352
Four	367
Five	278
Six	247

In run one, the phenotype found in the final population began at least as early as generation 67, about 15% of the final fitness. However, while the basics are there, the behavior is sluggish in comparison to the final version. At 10% of final fitness, generation 53, the starting configuration is the same, but simply falls over into a different configuration, which does not move. The first time sample at 5%, was at generation 25 and appeared to have all of the point-masses and some of the structure in place, but lack stability and collapsed down onto the ground with one point-mass held in the air. Run two finds its final form sometime after generation 46, 25% of the final fitness. At this point, the final structure is in place, but it does not move. Then, starting at the earliest sample at generation 17, each progressive sample up to generation 46 builds off an initial standing pyramid. As time progresses the structure approaches the final structure but continues to collapse into secondary configurations until sometime after generation 39, 20% of the final fitness, the last point-mass is added in the correct position to keep the final structure intact.

For the runs that appear not to stagnate, runs three and four have a much longer history. To begin, the MRCA for both has the same general structure and behavior as the final phenotype, but slightly less stable structure. Run three begins at 5% of the final fitness, generation 108, as a simple pyramid with a chain of unused point-masses hanging from it. Then up through generation 168, a basic stacked pyramid structure had evolved that falls over to form the basic final structure. From generation 168 to 230, the final structure is in place, but cannot move because it can barely get its front point-masses off the ground. At generation 322, about 50% of the final fitness, the individual has started moving slightly through expansion and contraction, and has started picking up its leading point-masses.

Run four has fewer noticeable intervals of change, but with a long history. The initial pyramid form at generation 20 continues to increase in complexity until it is close to the final structure before generation 175, 45% of final fitness. At this point, it has begun to move by picking up its leading points and rocking back and forth. This behavior is unstable and moves into a secondary behavior for a short while before getting back into it primary behavior. Interestingly, the secondary behavior is similar to the final behavior. Finally, after 50% of the final fitness has been reached at generation 218, it has evolved the general structure and behavior of the final population.

Due to the large final fitness of runs five and six, they have very short intervals of change. Run five has achieved 5% of the final fitness by generation 78, and has the basic phenotype without the leading point-mass and can move through expanding and contraction. By generation 99 and then 107, 10% and 15% of the final fitness, the leading point-mass has been added to the structure making it like the final structure. It

also moves using the final behavior, but cannot really jump. The rest of the evolution seems to focus on improving three aspects: getting into its leap configuration, leaping, and maintaining linear movement. In run six, the lineage has achieved the final structure by generation 83, 5% of the final fitness, but can not move. However, by 10% of the final fitness, generation 94, it has attained the final phenotype.

The evolution of the initial structures is interesting and may indicate the path of future evolution. Any new behavior will most likely occur by adding complexity and altering the structure. It appears unlikely that runs will have high variation of structure and behavior, but transitions between distinct behaviors are more likely. These comparisons have also shown that the runs tended to lock in a structure and behavior early, and the evolution that follows, selects for improvement on that phenotype.

5.2.3 Alternative Method For Measuring Genetic Diversity

With the extra knowledge of structure and behavior variation in both the final generation and along the lineage, these case studies afford me the opportunity for further examination of genetic similarity. In Chapter 3, genetic diversity measures, such as genetic richness (the count of unique genomes) indicated near maximal diversity. Additionally, those measures showed little genetic similarity between individuals in a given population or between the MRCA and the final population. However, the data examined in this chapter suggest that while there is variation in structure and behavior, the variation is not that dramatic. One potential reason for the discrepancy between previous measurements and phenotype diversity is the weak selection and the large percentage of neutral mutation accumulation over time (on average, 70% of mutations in the final generation are neutral). These two factors can add a lot of extra instructions that

translate into increased genetic difference, but have no affect on the resulting phenotype or fitness.

I use the following method to address the discrepancy between the high genetic diversity, low genetic similarity within a population, and similarity in structure and behavior examined in this chapter. I knock out as many neutral instructions from a genome as possible, which can be considered condensing the genome. These condensed genomes are then used to determine the genetic richness of the final population, the genetic similarity within the final population, and the similarity between MRCA and the final population. I then compare these values to their counterparts in Chapter 3.

I chose a simple method for condensing genomes. The genome is traverserd linearly, each instruction is individually removed and the resulting phenotype is measured. If the fitness value of the new phenotype is the same as the original fitness, the current instruction is marked as neutral. After all of the instructions have been tested, all of the neutral instructions are removed from the genome.

This genome condensing method produces only an approximation of the minimal genome for a given individual. Any difference from the true minimal genome is the result of epistatic effects, which are not considered in this study. However, epistatic effects were found in on 0 to 15% of the individuals for any given run.

For genetic richness, the count of unique genomes, the value was much lower for the condensed populations than what was found in Chapter 3. Due to the approximation in the condensing method, the richness may be lower than the true value of the minimal genome. The reason is that the approximation would remove extra instructions, increasing the chance of identical genomes. Table 5.2 shows the comparison between

genetic richness from Chapter 3 and the new method. Also, phenotypic richness, in terms of fitness, is also given as a point of comparison.

Table 5.2. Richness measures for all six runs. Column two is the original genetic richness from Chapter 3. Column three is the genetic richness using condensed genomes, and column four is the phenotypic richness in terms of fitness.

Run	Original	Condensed	Phenotype (Fitness)
One	451	242	171
Two	457	205	155
Three	442	301	237
Four	431	276	190
Five	431	319	248
Six	464	274	226

The Table 5.2 indicates that richness measures of Chapter 3 are heavily dominated by the accumulated neutral instruction. It is also interesting to note that the genetic richness for condensed genomes is still much higher than phenotypic richness. This means that different genomes have different phenotypes with the same fitness. It would be an interesting follow up to determine how much of this difference is due to genomes that map to phenotypes with low fitness.

My next comparison is based on the similarity between each pair of genomes within the final population. Table 5.3 shows the results from Chapter 3 versus the condensed genomes for both similarity and dissimilarity measures. The similarity measure is a ratio of the difference between the maximum genome length and the edit distance over the minimum genome length. The dissimilarity measure is the ratio of the edit distance over the maximum genome length. For more details see Sections 3.1 and 3.2 from Chapter 3.

Table 5.3. The dissimilarity and similarity measures are given for each run along with their 95% confidence interval.

Run	Original Dissimilarity	Condensed Dissimilarity	Original Similarity	Condensed Similarity
One	0.779 ± 0.017	0.368 ± 0.016	0.238 ± 0.017	0.691 ± 0.017
Two	0.806 ± 0.015	0.257 ± 0.011	0.210 ± 0.015	0.775 ± 0.010
Three	0.562 ± 0.019	0.454 ± 0.016	0.458 ± 0.019	0.594 ± 0.016
Four	0.798 ± 0.015	0.619 ± 0.016	0.219 ± 0.015	0.436 ± 0.016
Five	0.823 ± 0.017	0.677 ± 0.017	0.194 ± 0.017	0.369 ± 0.017
Six	0.761 ± 0.015	0.454 ± 0.017	0.258 ± 0.016	0.604 ± 0.016

Next, the similarity and dissimilarity are measured between the MRCA and the final population, and compared to what was found in Chapter 3. Both of these measures are shown in Table 5.4.

Table 5.4. The dissimilarity and similarity measures are given for each run including their 95% confidence interval.

Run	Original Dissimilarity	Condensed Dissimilarity	Original Similarity	Condensed Similarity
One	0.899 ± 0.002	0.611 ± 0.006	0.166 ± 0.003	0.430 ± 0.006
Two	0.794 ± 0.004	0.182 ± 0.009	0.223 ± 0.005	0.837 ± 0.009
Three	0.853 ± 0.003	0.654 ± 0.007	0.193 ± 0.004	0.374 ± 0.008
Four	0.759 ± 0.004	0.539 ± 0.010	0.273 ± 0.004	0.503 ± 0.010
Five	0.863 ± 0.002	0.836 ± 0.005	0.190 ± 0.003	0.182 ± 0.006
Six	0.790 ± 0.002	0.507 ± 0.010	0.337 ± 0.004	0.521 ± 0.008

Clearly, major differences are possible between measurements using condensed verses non-condensed genomes. Given the strong, consistent measurements of high genetic diversity, which seemed unreasonable to be maximized, the use of condensed genomes for these measurements may prove more reliable and informative. This makes sense because you are strictly using the parts of the genome that are important for selective purposes. The neutral instructions can be considered random because while

they may hitchhike, it has been shown that they allow mutations to accumulate, which drives apart the common neutral set of instructions.

By looking at the information provided using the condensed genomes, it is apparent that there is more similarity between genomes in the final population for the runs that have little improvement in fitness. In run six, it is only in the last generations that it breaks free of its stagnating fitness. It is also interesting that run three has a higher similarity when its fitness has been on the rise for the majority of the run. However, it is unclear to what extent, if any, there is a correlation between fitness and similarity, which may provide interesting information in the future. For now, it is interesting enough to note the variation in similarity between the six runs, and the variation among the observed phenotypes.

Similarity between the MRCA and the final population is also quite different when using condensed genomes. Run two appears to have almost inverted its quantity of similarity, while run five is nearly unchanged. These values seem to correlate with two factors observed among the samples individuals. The first factor is the number of individuals in the final population who exhibit the exact same structure and behavior as the MRCA. The second factor is the evolutionary difference of the MRCA to the common and most-fit individuals in the final population, and the generational difference between the MRCA and approximate first generation to exhibit the final structure and behavior along the lineage. Most likely, the issue of the loss of structure and behavior variation is a combination of many factors, and that information provided here show some relationships but nothing stands out as a primary indicator.

5.3 Conclusions

The six most-fit individuals all demonstrated different structure and behavior, but appear to share many common themes. The most significant similarity is the tendency to have every point-mass connected to nearly all of the others. There are also a large number of spike components to each body. From comparing various individuals, these two commonalities seem to contribute the most to improving fitness, but most likely the contributions are gained by adding stability to the structure. For behavior, the primary method of movement centers around picking up points off the ground, moving them forward, and then dragging the rest of the body forward through contraction.

Each of the six case studies demonstrated that a structure and behavior become fixed in the population early in the evolutionary process. Then, selection pressure drives the population toward improving that structure and behavior. Improvements are often accomplished through the alteration of the structure to cause changes in an individual's rhythm. Also, changes in structure and behavior did occur early in each run and toward the end of run six. Those changes to structure and behavior primarily arose through an increase in structural complexity.

After viewing the lack of structure and behavior variation in the final population and along the lineage, it is apparent that the genetic diversity and phenotypic variation measures from Chapter 3 are not good indicators of structure and behavior variation. Instead, they most likely indicate the variation in improvements to the structure and behavior that is fixed in the population. The usefulness of these indicators for phenotypic variation seems to improve when you switch from using the whole genome to a condensed genome. However, genome condensation may not be an appropriate

alternative for every type of evolutionary system or language, but these results do suggest possible confounding effects in the simple measures. In the end, it is clear that there are many additional factors that affect the variation of structure and behavior. Those factors addressed here show that using similarity information in conjunction with fitness can provide some indication of phenotypic convergence and stagnation. Structural and behavioral stagnation can be identified by comparing a population to its MRCA, while convergence can be identified by comparing condensed genomes..

Chapter 6 Conclusions and Future Work

I have examined the WhirlingDervish system in light of other 3D morphology and behavior systems and their research. WhirlingDervish presents an alternative variation for this type of evolutionary system in order to decrease run time and increase population size allowing for experiments to be performed that can answer fundamental questions about evolutionary dynamics. However, due to the difficulty of analyzing the evolutionary results, especially in terms of behavior, it is hard to compare these systems. In an attempt to gain common ground between these systems, I examined two aspects common to all of them: convergence and stagnation. It is my hope that these two aspects along with the work by Komosinski et. al. (2001a) in morphological classification will lead to further improvements in the behavior and system analysis.

In my opinion, one of the most beneficial additions to studying the evolution of morphology and behavior would be a way to classify changing morphology in terms of behavior, preferably with an automated process rather than one that involves visual analysis. Having the ability to classify individuals based on behavior can provide many advantages such as a quick way to identify good solutions or significantly reduce the set of solutions from which to sample good solutions. It can also be useful in measuring individual and system complexity. Also, measuring diversity and variation in the previous chapters gave a rough idea of convergence and stagnation in WhirlingDervish, perhaps behavior classification would have given different or more precise information about the system by characterizing diversity in terms of the system goal.

Current research on behavior classification in 3D evolving systems is limited at best. Therefore, I recommend the work of Walter F. Bischof and Terry Caelli as a potential starting point. Their work centered on the use of a basic temporal pattern classification system that utilizes movement data from key structural points on a moving object. It then utilizes the temporal data for classification. In one situation, they used their system to determine if a person was constructing one of three different designs out of straight and bent tubes by monitoring the movement and speed of the hands (Bischof and Caelli, 2001a). Another situation determined if a person was lifting either a heavy or light box and if they were using their legs or their back for lifting (Bischof and Caelli, 2001b).

Looking once again at convergence and stagnation, an alternate population and fitness assignment model may prevent, or at least stave off, convergence and stagnation in WhirlingDervish, which should ultimately improve both the quality and rate of finding good solutions. One such approach is Hierarchical Fair Competition (HFC; Hu, 2004), which addresses finding global optima and how to continue searching if an algorithm has converged upon a local optima by maintaining diversity. HFC uses a population model based on a hierarchy of isolated populations that conform to a specific fitness range. Using a promotion/demotion scheme provides a way to move individuals between levels and potentially increase population diversity. It attempts to address continuous evolution by creating new low fitness solutions that can rise through the ranks and are not immediately wiped out because their only competition are the others at the same population level (Hu, 2004). While this thesis has shown that convergence and

stagnation did not occur using the specified experimental design, this approach may prove helpful in experiments with longer time frames.

Another interesting avenue of research is self-replication and developmental models. The WhirlingDervish system would make an excellent platform for this type of research since its phenotypes are heavy based on physics models. Both of these avenues would work together to allow offspring to be born in real time. Normally, the offspring are created instantaneously in a vacuum. This research could provide insights into many different areas of evolution including developmental stages, the effects of historical contingency, and potentially the introduction of competition as a form selective pressure rather than a static fitness function.

The study of genetic language for constructing morphology is a primary research topic for many of the evolutionary systems for 3D morphology and behavior. In the course of creating WhirlingDervish and writing this thesis, I conducted a series of experiments using the genetic language in WhirlingDervish. These experiments examine the use of common programming structures such as loops, conditions, and jumping to labels and how they affect WhirlingDervish results. However, due to time constraints, I was unable to add them to the thesis. I hope to publish these experiments at a future time.

Finally, the use of diversity measures in WhirlingDervish toward the study of the Cambrian Explosion theory may be an exciting and useful application of these types of evolutionary systems to evolutionary biology. The theory hypothesizes that all of the major body plans in existence evolved within a short span of time. As time has gone on, no new general body plans have been produced. It would be interesting to see if this effect occurs in WhirlingDervish and if so, why. Basically, I would be looking for

kernels of behavior and dynamic structure that evolved early in a run and are maintained over time. I would then need to show that while the behavior and dynamics have changed, their abilities are still based primarily on the earlier form. Of course, this study is also limited by the inability to classify behavior.

Appendix A Instruction Sets

The WhirlingDervish instruction set consists a default and extended version. The extended instruction set utilizes all of the instructions from the default set plus some additional ones. By using the instruction_set.cfg configuration file, a user can turn on or off any given instruction for a run.

Instruction Name	Instruction Description
add	Add the parameter to the top of the current stack
add-line	Using the parameter as an offset from the location of the current point-mass, a spring is created between the current point-mass and the point-mass closest to the new position. This could even be the current point-mass itself. After the creation of the spring, the current point-mass becomes the one that was closest to the new position.
div	Divide each component of the top of the current stack by each respective component of the parameter. Any divide by zero is set to zero.
merge	A point-mass is found that is closest to the location of the current point-mass plus the offset vector represented by the parameter. Nothing happens if an attempt is made to merge a point-mass with itself. Otherwise, a new point-mass is created whose mass is the average of the two point-masses at a location halfway between them. Then, for each spring connected to each of the old point-masses, a duplicate spring is connected to the new point-mass. Then the two original point-masses are removed from the individual.
mod	Same as divide, except modulus instead of divide
move-to	Finds a point-mass closest to the location of the current point-mass plus the offset represented by the parameter, and sets that point-mass to be the new current point-mass.
mult	Multiply each component of the parameter to each respective component of the top of the current stack
nop	Does nothing
pop	Pop the value off the current stack. If the stack becomes

Instruction Name	Instruction Description
	empty, a vector of all zeros is pushed onto the stack.
push	Push the parameter onto the current stack
push-1	Push a vector consisting of the first component of the parameter onto the current stack
push-2	Push a vector consisting of the second component of the parameter onto the current stack
push-3	Push a vector consisting of the third component of the parameter onto the current stack
remove-line	Using the parameter as an offset from the location of the current point-mass, the constructor will move to the point-mass closest to the new location and remove the spring between them. If no spring exists between the point-masses, the constructor doesn't move.
remove-point	Removes the current point-mass and moves to a point-mass that is closest to the current location plus the offset represented by the parameter
rotate	Changes the orientation of the individual builder: x->yaw, y->pitch, and z->roll
set-point-parameters	Sets the point-mass default data to be the parameter, but currently does nothing
set-spring-constants	Sets the spring constants to be the value of the parameter, but currently does nothing
set-spring-fluctuations	Sets the spring fluctuation data to be the value of the parameter, where each component is constrained between -1 and 1
shift-data-left	Shift the vector components on the top of the current stack to the left. x->z , y->x , z->y
shift-data-right	Same as shift-data-left but shifts the data to the right
split	The parameter is a vector, which is first normalized, then scaled by the dilation state variable of the constructor object. The modified vector is then used as an offset from the location of the current point-mass to create a new point-mass at that location. This new point-mass is connected to the current point-mass by a single spring. No spring is created if there was no originating point-mass, which means the constructor object's location would have been used as the starting location. The new point-mass then becomes the current point-mass.

Instruction Name	Instruction Description
split-dup	Same as the split instruction but creates additional springs. For each spring connected to the originating point-mass, a duplicate spring is created with the new point-mass. In addition, the spring between the current point-mass and the new point-mass is still created.
sub	Subtract the parameter from the top of the current stack
switch-stack	Switches the current stack to one of the four stacks where the parameter can only be the name of one of those stacks

Table A.1. Default Instruction Set

Instruction Name	Instruction Description
else	If this instruction does not fall between an if and end instructions, it is ignored. If the if instruction fails, the instructions between the else and end are executed, otherwise the instructions between the if and else are executed.
end	An instruction denoting the end of a control block, for both if and loop instructions.
if	An if instruction must have a terminating end instruction, or it is ignored. The instructions within the if block are only executed if the if is successful. The if statement parameter is used to determine the success of the instruction using the following logic if x >= y then true if x <= y then true if x == y then true
jump	Finds the appropriate label, push the current constructor object's state onto a special stack and move to the appropriate location in the genome to continue executing instructions.

Instruction Name	Instruction Description
label	A label is similar to a function in that the instructions between it and the next one can only be executed by a jump to that label. A label denotes a specific location in the genome and its parameter is an id used by the jump instruction to move the thread of execution to that location. If more than one id exists, the last label is the one that is used. If a label is the first instruction, a jump to that label is automatic. Otherwise, you can never execute a label's instructions unless a jump is used. Once the last instruction in a label is executed the construction object state stack is popped and its state restored. Once the last state has been popped off the stack, the execution is concluded.
loop	The loop instruction executes the block of instructions between itself and the end instruction, at most ten times. The loop is ignored if there is no terminating end instruction. The condition logic is the same as the if instruction, and the increment value utilizes the z component in the follow fashion. -z -> if x >= y then true +z -> if x <= y then true 0 -> if x == y then true
partial	The parameter is used to set the partial state in the construction object, which is used by split-partial.
pop-d	Restores the construction object's orientation and position from the top of the orientation stack. If the stack is empty, this instruction does nothing. The current point-mass is set to the point-mass nearest the restored location of the construction object.
pop-line-d	Same as pop-d, except a spring is created between the previous point-mass and the new point-mass
push-d	Pushes the construction object's orientation and position onto the orientation stack.
set-dilation	The dilation vector is set to be the parameter. This is used in split instructions to scale the offset vectors.
split-dup-oriented	Same as the split-orientation instruction with the creation of springs like the split-dup instruction.

Instruction Name	Instruction Description
split-oriented	This is the same as a split, except the forward vector of the orientation of the construction object is used instead of the parameter as the offset. The parameter is then used as the equivalent of the rotate command using the split-oriented parameter as the rotate parameter, which occurs prior to the creation of the new point-mass.
split-partial	This is the same as a split-dup with an exception. Instead of making duplicate of all of the springs, the parameter is used to determine a percentage of springs to create. Consider the springs as an array, x is the percentage of springs to duplicate from the start of the array, y is the percentage around the middle of the array, and z is the percentage from the end of the array. The percentages can overlap, but a spring can never be replicated more than once.
split-partial-oriented	Same as a split-partial, but also utilizes the orientation aspect of split-oriented.
top-d	Same as pop-d, except the data is not popped from the stack.

Table A.2. Instruction Set Extensions

Appendix B The Construction Object

The construction object is the external mechanism for replicating individuals. It consists of two parts: a list of functions that correspond directly to the instruction set and a set of vectors comprised of three real numbers that hold state information. The object begins work when given a genome by linearly passing from the first instruction to the last. When an instruction is executed, the corresponding internal function is called. The functions do all of the work in generating the body by using a combination of internal state information and the instruction parameter. Once the last instruction has been executed, the body is complete and the genome and body are released back to the system.

There is a specific set of state variables available for use by the instructions that manipulate the construction object during execution. The use of state variables allows a reduction in instruction complexity by moving necessary information from the instruction's parameters into the construction object. Moving information into the object requires that new instructions be added to the instruction set specifically to manipulate the new states. Thus, a trade-off occurs between complex instructions and the need to execute multiple instructions to accomplish the same task. However, manipulation of the state variables is not necessary as each variable begins in a default state, and once the state is set, future instructions will reuse that same state.

The state variables available for use by the instruction set are four stacks of vectors, spring constants, spring fluctuation data, point-mass properties, gravity, dilation, partial data, and the construction object location and orientation. The stacks are available for push and popping vectors for later use and for the results of mathematical instructions. The spring constants, spring fluctuation data, and point-mass properties are

each a vector describing the properties of the respective physical component when it is added to the body. The gravity vector is constant and normalized, and can only be used by an instruction for the purposes of direction. The dilation and partial data are scaling effects upon a new point-mass. The location and orientation of the construction object can be changed through instructions, but are mostly used as direction vectors.

Finally, the construction of the body is accomplished by executing the genome, which is a simple program or script that drives the construction object. Each body begins empty and the first point-mass created always begins at the origin. As each new point-mass is added to the body, the construction object keeps track of which point-mass it is working on currently. The current point-mass is a key factor in body construction because of the relative nature of the build order to that current point-mass. The relative nature is based on the process where each new point-mass that is created uses the instruction parameter as an offset from the current point-mass.

BIBLIOGRAPHY

Adami C. *Introduction to Artificial Life*. New York: Springer-Verlag, 1998.

Angeline PJ, Pollack JB. Coevolving High-Level Representations. In Langton CG (ed). *Artificial Life III*, pp 52-72. Menlo Park, CA: Addison-Wesley, June 1994.

Bischof WF, Caelli T. On the learning of complex movement sequences. In Arcelli C, Cordella LP, Sanniti di Baja G (eds). *Visual Form*, pp 463-472. New York: Springer, 2001a.

Bischof WF, Caelli T. Learning complex action patterns with CRGst. In Singh S, Murshed N, Kropatsch W (eds). *Advances in Pattern Recognition (IACPR 2001)*, pp 280-289. 2001b.

Bongard JC, Lipson H. Once More Unto the Breach: Co-evolving a robot and its simulator. *Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems(ALIFE9)*, pp 57-62. 2004.

Bongard JC, Pfeifer R. Evolving Complete Agents Using Artificial Ontogeny. In Hara F, Pfeifer R (eds), *Morpho-functional Machines: The New Species (Designing Embodied Intelligence)*, pp 237-258. Springer-Verlag, 2003.

Bongard, JC, Pfeifer R. Repeated Structure and Dissociation of Genotypic and Phenotypic Complexity in Artificial Ontogeny, In Spector L (eds), *Proceedings of The Genetic and Evolutionary Computation Conference, GECCO-2001*, pp 829-836. San Francisco, CA: Morgan Kaufmann, 2001.

Bossert W. Mathematical Optimization: Are There Abstract Limits on Natural Selection? In Moorehead PS, Kaplan MM (eds). *Mathematical Challenges to the Neo-Darwinian Interpretation of Evolution*, Philadelphia, PA: Wistar Institute Press, 35-46, 1967.

Burke EK, Gustafson S, Kendall G. Diversity in Genetic Programming: An Analysis of Measures and Correlation with Fitness. *IEEE Transactions on Evolutionary Computation*, 8(1):47-62, 2004.

Daida JM, Ward DJ, Hilss AM, Long SL, Hodges MR, Kriesel JT. Visualizing the Loss of Diversity in Genetic Programming. *Proceedings of the 2004 IEEE Congress of Evolutionary Computation*, pp 1225-1232. Portland, OR, Piscataway: IEEE Press, 2004.

Ekart A, Nemeth SZ. Maintaining the Diversity of Genetic Programs. In Foster JA, Lutton E, Miller JF, Ryan C, Tettamanzi A (eds). *Proceedings of the 5th European Conference on Genetic Programming*, pp 162-171. London, UK: Springer-Verlag, April 2002.

Fogel DB. An Introduction to Simulated Evolutionary Optimization. *IEEE Transactions on Neural Networks*, 5(1):3-14, 1994.

Hornby GS, Pollack JB. Creating High-Level Components with a Generative Representation for Body-Brain Evolution. *Artificial Life*, 8(3):223-246. 2002.

Hornby GS, Lipson H, Pollack JB. Evolution of generative design systems for modular physical robots. *IEEE International Conference on Robotics and Automation*, pp 4146-4151. 2001a.

Hornby GS, Pollack JG. Body-Brain Co-evolution Using L-systems as a Generative Encoding. *Genetic and Evolutionary Computation Conference*, pp 868-875. 2001b.

Hu J. *Sustainable evolutionary algorithms and scalable evolutionary synthesis of dynamic systems*. Doctoral dissertation, Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, 2004.

Komosinski M. The Framsticks system: versatile simulator of 3D agents and their evolution. *Kybernetes: The International Journal of Systems & Cybernetics*, 32(1/2):156-173, MCB University Press, 2003.

Komosinski M, Koczyk G, Kubiak M. On Estimating Similarity of Artificial and Real Organisms. *Theory in Biosciences*, 120:271-286, 2001a.

Komosinski M, Rotaru-Varga A, Comparison of different genotype encodings for simulated 3D agents. *Artificial Life Journal*, 7(4):395-418, MIT Press, 2001b.

Komosinski M. The World of Framsticks: Simulation, Evolution, Interaction. In Heudin JC (ed). *Virtual Worlds. Lecture Notes in Artificial Intelligence*, pp 214-224. 2000.

Komosinski M, Ulatowski Sz. Framsticks: towards a simulation of a nature-like world, creatures and evolution. *Proceedings of 5th European Conference on Artificial Life (ECAL99)*, pp 261-265. Lausanne, Switzerland: Springer-Verlag, September 1999.

Koza JR. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press, 1992.

Lipson H, Pollack J B. Evolving Physical Creatures. In Bedau MA, McCaskill JS, Packard NH, Rasmussen S (eds). *Proceedings of Artificial Life VII* (ALIFE7), pp 282-287. Portland, OR, 2000.

Magurran AE. *Measuring Biological Diversity*. Oxford, UK: Blackwell Science, 2004.

McPhee NF, Hopper NJ. Analysis of genetic diversity through population history. In Banzhaf W, Daida J, Eiben AE, Garzon MH, Honavar V, Jakiela M, Smith RE (eds). *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO99)*, pp 1112-1120. July 1999.

Ofria C, Wilke CO. Avida: A Software Platform for Research in Computational Evolutionary Biology. *Journal of Artificial Life*, 10:191-229, 2004.

Pollack JB, Hornby GS, Lipson H, Funes P. Computer Creativity in the Automatic Design of Robots. *LEONARDO*. 36(2):115-121. 2003.

Popovici E, De Jong K. Understanding EA Dynamics via Population Fitness Distributions. In Cantu-Paz E, et. al. (eds). *Proceedings of the Genetic and Evolutionary Computation Conference*, pp 1604-1615. 2003.

Ray TS. Evolution, complexity, entropy, and artificial reality. *Physica D*, 75:239-263, February 1994.

Rosca JP. Entropy-based adaptive representations. In Rosca JP (ed). *Proceedings of the ML95 Workshop: From Theory to Real-World Applications*. pp. 23-32. 1995a.

Rosca JP. An Analysis of Hierarchical Genetic Programming. *Technical Report 566*, University of Rochester, NY, 1995b.

Ruebsamen GD. *Evolving Intelligent Embodied Agents Within a Physically Accurate Environment*. Doctoral dissertation, Department of Engineering and Computer Science, California State University, Long Beach, CA, December 2002.

Sims K. Evolving Virtual Creatures. *Proceedings of SIGGRAPH 94*, pp 15-22. July 1994a.

Sims K. Evolving 3D Morphology and Behavior by Competition. In Brooks R, Maes P (eds). *ALife IV: Proceedings of the 4th Conference on Artificial Life*. pp. 28-39. MA: MIT Press, 1994b.

Soda Play, www.sodaplay.com and www.sodaplay.com/constructor/how/works.htm.

Taylor T, Massey C. Recent Developments in the Evolution of Morphologies and Controllers for Physically Simulated Creatures. *Artificial Life*, 7(1):77-87, 2001.

Taylor T. www.tim-taylor.com/research/embody.html.

Ventrella J. Sexual Swimmers: Emergent Morphology and Locomotion without a Fitness Function. *From Animals to Animats*, pg 484. MIT Press, 1998.

Ventrella J. Explorations in the Emergence of Morpholgoy and Locomotion Behavior in Animated Characters. In Brooks R, Maes P (eds). *Proceedings of the Fourth Workshop on Artificial Life*, Boston, MA: MIT Press, 1994.

Verlet L. Computer "*Experiments*" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules. *Physical Review*, 159:98-103, 1967.