To the Graduate Council:
I am submitting herewith a dissertation written by Jay Jay Billings entitled “Optimization of Cosmological Simulations with Artificial Intelligence”. I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Physics.

Mike Guidry, Major Professor

We have read this dissertation and recommend its acceptance:

Otis Earl Messer

John Z. Larese

Accepted for the Council:

Carolyn R. Hodges, Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)
Optimization of Cosmological Simulations with Artificial Intelligence

A Dissertation
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Jay Jay Billings
May 2008
Dedication

To my Fathers, he who art in Heaven and he who art in Marion, and to everyone who’s done took one for the hood.
I would like to acknowledge, first, the members of my committee for not only agreeing to take me as a student, but each for a very specific reason: Dr. Mike Guidry, my major advisor, for giving me this project, regular encouragement and guidance, an always-open office door, and his kindness to a student others were willing to ignore; Dr. Bronson Messer for sharing his unfathomable knowledge of computers and Astrophysics; Dr. John Larese for briefly jumping from Chemical Physics to Computational Astrophysics, letting me loose on his 102 node Beowulf cluster (thus enabling my speedy completion), and his very good personal advice.

I would also like to acknowledge the University of Tennessee’s Department of Physics and Astronomy for two and a half years of financial support and the Astronomy Education research funds provided by Dr. Mike Guidry. Counting the $16,500 yearly stipend, some summer funds, and the $9,000/semester out-of-state tuition costs, I am very grateful to acknowledge this contribution of nearly $90,000 to my education.

There are several acknowledgments I should make in reference to the software developed in this thesis. Thank you Dr. Joel Welling for that pre-release of Starsplatter2.0, Dr. Volker Springel for Gadget2, Dr. John Bullinaria for your excellent FFBP-Neural Network tutorial, Dr. Genevieve Orr for your online CS-449 Introduction of Neural Networks course, and, last but certainly not least, UT’s own Dr. Bruce MacLennan for his expert advice and guidance on Artificial Neural Networks that eventually led to the creation of Megatron.

There are some personal acknowledgments that I care to make. I’ll list them in case some of you are looking. ;)

**Viktor Chupryna - Nastoyka**
Elisha “Chainsaw” Feger - Master Debugger, Master of the Universe.

Dwayne John - Thanks for always hanging with me on those much needed trips to the candy shop. I know it must have gotten boring the 10,000th time or so, but the fuel of every thesis is caffeine and theobromine.

Shuhua Liang - Thanks for the many discussions that lead to the completion of Starscream and, also, always hanging with me on those 10,000 trips to the candy store.

Jafet Morales - Master Debugger of my genetic algorithm.

John Simonetti - Thanks for the fish!

And finally, Paige Landry, my beautiful girlfriend, thank you for loving me, watching out for me, and taking such good care of me. I love you and I owe you so much more credit than I can give you here for all the help and love you showed during the completion of this work. You really are the best cook of your age. :P
Abstract

Galaxy collisions are an important part of the large-scale structure of the Universe and an important catalyst for intragalactic processes like star formation. Therefore, realistic models of these interactions are an important part of any theory that plans to accurately describe the evolutionary processes of the Universe and, given the size of the problem, efficient computation and data analysis are key. This dissertation presents a proof-of-concept that an artificial intelligence suite, nominally composed of a genetic algorithm and neural network, can optimize the search of the galaxy collision parameter space. Furthermore, this dissertation discusses the possibility that this method can be used for any large problem dependent on a large number of tunable parameters.
Contents

1 Introduction 1
   1.1 Galaxies and their Interactions .......................... 3
       1.1.1 Galaxies in brief ........................................ 3
       1.1.2 Colliding Galaxies ...................................... 4
   1.2 Genetics ...................................................... 5
       1.2.1 Populations and Individuals .......................... 5
       1.2.2 Crossover and Mutation ................................. 5
       1.2.3 Generations and Fitness ................................. 6

2 Modeling Galaxy Collisions 7
   2.1 The N-body Method ............................................ 7
       2.1.1 Gravity Trees ............................................. 8
   2.2 Model Galaxies ................................................. 8
       2.2.1 Exponential Disks ....................................... 10
       2.2.2 Dark Matter Halos ...................................... 12
       2.2.3 Disk in a Halo? ......................................... 13
       2.2.4 Initial Conditions ...................................... 15

3 Feed-Forward Back-Propagating Artificial Neural Networks 18
   3.1 The Artificial Neuron ......................................... 18
       3.1.1 Activation Functions: sigmoid vs. tanh ................. 20
   3.2 Error Back-Propagation ....................................... 21
   3.3 Megatron ..................................................... 23
3.3.1 Test Case: To Catch a Terrorist ................. 23

4 Genetic Algorithms ................. 25
4.1 Parts of a Genetic Algorithm ................. 25
4.1.1 More on Crossover and Mutation ................. 27
4.1.2 Maximizing Functions ................. 28
4.2 OptimusPrime ................. 28

5 Results and Conclusions ................. 31
5.1 Results: Transforming Galaxy Collisions ................. 31
5.1.1 Galaxies ................. 32
5.1.2 Training ................. 32
5.2 Conclusions: The Future ................. 35
5.2.1 Galaxy Collisions: Starscream ................. 35
5.2.2 Pattern Recognition: Megatron ................. 36
5.2.3 Parameter Sweeping: OptimusPrime ................. 37
5.3 Other Codes ................. 37

Bibliography ................. 39

Vita ................. 43
List of Figures

1.1 Hubble Tuning Fork Diagram ........................................ 2
1.2 Examples of Popular Colliding Galaxies ............................. 6

2.1 Speedup using a Tree Algorithm ..................................... 9
2.2 Rotation Curve of NGC6503 .......................................... 11

3.1 Examples of Neural Networks in Fiction and Reality ............... 19
3.2 Pictures of Artificial Neurons ......................................... 20
3.3 Sigmoid Family Functions .............................................. 21
3.4 Results of Terrorism Test Case ....................................... 24

4.1 2D Gaussian ............................................................. 26
4.2 Graph of \( \frac{\sin(x)}{x} + \frac{\sin(y)}{y} \) ........................................ 29

5.1 Galaxy Collision 1 ..................................................... 33
5.2 Galaxy Collision 2 ..................................................... 34
5.3 Processed Galaxy Collision Images ................................. 35
Chapter 1

Introduction

Perhaps some of the most beautiful objects in the night sky are galaxies. They hold the position of “Majestic Heavenly Bodies” in our minds and even the Millennium Falcon flies into one at the end of *The Empire Strikes Back*. For astronomers and astrophysicists, they are just as interesting to study as they are to view and collisions between two galaxies are better than Thanksgiving dinner.

Colliding galaxies have been studied for roughly ninety years, starting with Lundmark and Lindblad around 1920 and theoretically modeling their tidal interactions is not new, [Struck, 1999]. Typically one requires a computer, an N-body code, and some knowledge of galactic dynamics. The computer does not even have to be digital: Erik Holmberg showed that spiral arms can appear from tidal interactions using seventy-four light bulbs and a photometer, [Holmberg, 1941].

The importance of studying these collisions appears when one considers the history of extragalactic astronomy and the dynamical effects of two objects over a trillion times as massive as the Sun. The former is a result of the Edwin Hubble’s “tuning fork” diagram, see Fig. 1, and the sheer number of galaxies classified as *peculiar* because they did not match the morphologies described by his fork, [Struck, 1999]. The latter comes from increased star formation in colliding galaxies, [Toomre and Toomre, 1972], active galactic nuclei, and similar phenomena. The plot thickens when dark matter and the large scale structure of the Universe are considered.

While it is certainly possible to gauge the accuracy of theoretical models with the eyeball
test, it would also be useful to quantify the degree of accuracy, or the *fitness*, of some result. Additionally, it would be useful to determine the values of any tunable parameters that provide these *fit* results if they produce objects reminiscent of observations. The obvious way to do this is to hire and train an undergraduate to run through every combination of the tunable parameters, meticulously cataloging each result along the way, and test the student’s classification skills when the job is finished for control. However, if there is a shortage of either undergraduates or funding, there exists an alternative: an Artificial Intelligence.

This thesis will provide, at least, a proof-of-concept that artificial intelligence can accomplish this task. Specifically, this thesis will present an artificial intelligence suite that varies the tunable parameters of collision-less N-body simulations and compares the results to observed collisions. The argument will be made that a *genetic algorithm*, a code based on the principles of Evolution, can “meticulously” vary the parameters of the N-body code Gadget2 and that an *artificial neural network* trained on images of observed collisions can reduce the accuracy problem to a virtual form of the eyeball test, simply a pattern
The following sections will provide a brief introduction to galaxies, specifically the classifications specified by the Tuning Fork diagram and examples of collisions, and a brief refresher of genetic evolution. The following chapters will outline in detail galaxy collisions, genetic algorithms, artificial neural networks, the combination of the three and the results.

1.1 Galaxies and their Interactions

1.1.1 Galaxies in brief

There are a few galaxies visible with the naked eye, namely the Magellanic Clouds and Andromeda, the latter of which was discovered over a thousand years ago, [Fix, 2001]. Galaxies were originally thought to be no different than planetary nebulae, until Hubble discovered Cepheid variables in Andromeda in 1923, [Fix, 2001]. This is highly significant since Cepheid variables are standard candles and, therefore, provide accurate measurements of distance on an intergalactic scale.

Galaxies are historically classified within three different types: Ellipticals, Spirals, and the aforementioned Peculiars, also called Irregulars. The relationship between the first two is established on the Tuning Fork diagram and Irregulars are simply anything not of the previous two types. (This scheme is due to Hubble, as was previously mentioned.) Elliptical galaxies are ellipsoids varying in eccentricity from zero to nearly one and are mostly uniform. Spiral galaxies are disks with arms that wrap around a central core. Not all spirals are the same, with many galaxies exhibiting tightly bound arms while others have loosely bound arms. Some spiral galaxies also have bars across their center, (designated notationally by the addition of a “B”), and the size of a bar is also different between galaxies.

The work described in this thesis concentrates on spirals because they are the most observationally abundant type. Spirals are composed of 10-100 billion stars that have a cumulative mass around $10^{11}$ solar masses [Binney and Tremaine, 1987]. This visible matter represents less than ten percent of the total mass of a galaxy and there exists a large amount of dark matter that only interacts gravitationally. The largest evidence for dark matter lies in the measured rotation curves of galaxies. In short, the circular velocity for a given
galaxy stays constant at large distances from the central core, but theory predicts that it should rapidly decrease. This points to something unseen or, rather, something that does not interact electromagnetically, but has a very large mass.

1.1.2 Colliding Galaxies

Colliding galaxies represent about 1-10% of all observed galaxy systems, but studies suggest that most galaxies have experienced at least one significant collision in their past, [Struck, 1999]. If the mass of each galaxy is on the order of $10^{12}$ solar masses, including their individual dark matter halos, and they collide at roughly 300 km/s, then the kinetic energy of the collision is

$$E_{\text{kin}} = \frac{mv^2}{2} = \frac{3.978 \times 10^{45} \text{g} \times (3.00 \times 10^7 \text{cm/s})^2}{2} = 1.79 \times 10^{60} \text{ergs}$$

[Struck, 1999] also makes a very nice point about this energy: it is on the order of one billion supernova explosions.¹

The most popular question about galaxy collisions can also be answered: “If the Galaxy collides with another galaxy, won’t we all die?” The answer is no, considering the mean free path. Let $n$ denote the number density of stars in a galaxy, $\sigma$ the surface area of the Sun, and $\lambda$ the mean free path. Also assume that the radius of a galactic disk is about 50 kpc with a disk thickness of $\sim 0.5$ kpc and that the number of stars in a galaxy is roughly $10^{11}$. Therefore,

$$n = \frac{10^{11}}{V} \sim \frac{10^{11}}{hr^2} = 2.57 \times 10^{-57} \text{cm}^{-3}$$

$$\sigma = 4\pi r^2 = 8.75 \times 10^{22} \text{cm}^2$$

$$\lambda = \frac{1}{n\sigma} = 1.35 \times 10^{12} \text{kpc}$$

and it is immediately obvious that a stellar collision is improbable. This does not exclude the possibility that a tidal disturbance either ejects the Sun from the Galaxy or throws it into the supermassive black holes at the center of the merger, though.

¹Although this is only true in terms of visible matter. The neutrino luminosity of a supernova is one hundred times larger than that of the visible matter, [Guidry, 2007]. Therefore, this should only be taken as an estimate, accurate to within an order of magnitude or so.
There are some very well-known examples of galaxy collisions, shown in Fig. 1.2. These three systems all have long, extended tidal tails. They will be featured thoroughly in a later discussion.

1.2 Genetics

The definitions of some key words in genetics and the theory of Evolution are covered here not so much for educational reasons, but to clarify how they are used in this thesis. Evolution is simplified here in the sense that there are no predators and survival is dependent only on the desirability of an individual for mating.

1.2.1 Populations and Individuals

Populations will be defined as groups of individuals and are the equivalent of a mathematical set. Individuals are comprised of a number of traits that define their identity in the population, i.e. - individuals have a specific phenotype defined by their constituent genes.

1.2.2 Crossover and Mutation

Individuals in a population are allowed to exchange genetic material via a crossover. In the real, biological world, crossover happens via breaking and reunion in genetic material and crossover points for a chromosomal pair are determined chemically, [Klug and Cummings, 2000]. Although there will be a detailed discussion in chapter 3, it is important to note that the number of crossover points is determined a priori in this work and the positions on the chromosome are picked randomly.

Mutation is a random process that can affect either a gene or an entire chromosome. It can change the “value” of a specific gene or rewrite an entire part of a chromosome, [Klug and Cummings, 2000]. Generally, crossovers happen more frequently than mutations, but because they can drastically change the nature of an individual, and consequently the future population, mutations are just as important.
Figure 1.2: Examples of a few collisional systems. (a) NGC 4038/9 “The Antennae,” 15’ x 15’, north up (b) NGC 4676 “The Mice,” 5’ x 5’, north up (c) NGC 7252, 7’ x 7’, north up.

1.2.3 Generations and Fitness

Individuals in a population mate and produce offspring. The fittest members are the most likely to mate and fitness is determined by the degree to which a particular individual can survive in its environment. After a subset of the population produces offspring, a new generation has started and when those offspring mate, whether with each other or with an older member, another generation will start.

It is by this process, fit members of a population creating the next generation with different genetic material because of crossover and mutation, that populations evolve.
Chapter 2

Modeling Galaxy Collisions

This chapter will review the theory behind disk and dark matter halo models and provide a discussion of the approach taken to create tidal interactions between two model galaxies.

2.1 The N-body Method

The *N-body Method* is a technique used to calculate the forces on an object due to all other objects in the system. In the case of galaxy collisions, the force on a particle is due to gravity and the other objects are its neighboring stellar-type particles. Since galaxies are, for the most part, classical in nature, the gravitational force is Newtonian and the equation of motion for a given particle *i* is, [Aarseth, 2003]

\[
\ddot{r}_i = -G \sum_{j=1; j \neq i}^{N} \frac{m_j (r_i - r_j)}{|r_i - r_j|^3} \tag{2.1}
\]

There is not an exact solution for the above equation and it is necessary to find a solution numerically for *i* > 2, (or in the case that one of the masses is very small compared to the others, *i* > 3), given the initial masses, velocities, and positions. An advanced description of how to do this computationally is given in [Aarseth, 2003], but in section 1.4 Aarseth provides this introduction,

1.) Advance the velocity and coordinate data by one time-step using an explicit integration method.
2.) Reevaluate the equation of motion.

3.) Calculate the velocity and position via

\[ v_i(t) = F_i \delta t + v_i(t_0) \]  
\[ r_i(t) = \frac{F_i \delta t^2}{2} + v_i(t) t + r_i(t_0) \]

4.) Repeat until a certain value of time.

Although the above method is conceptually simple, the savvy reader will note that such a method takes \( \sim O(N^2) \) calculations, which is computationally expensive for large \( N \). See figure 2.1.

### 2.1.1 Gravity Trees

An approximation to the pure N-body method is the *Hierarchical Tree Method*, first introduced by Barnes and Hut in 1986. In this method, only the forces for the nearest neighbors are calculated by the direct summation method given above. Distant particles are taken as part of a mass distribution in some given cell, determined by recursively dividing large cells until some condition is satisfied, [Hernquist and Katz, 1989, Aarseth, 2003]. Each cell is allowed to contain only one particle and the space is divided until all cells are counted. In modern codes, the contribution of these distant particles to the total force is determined by finding a multipole expansion for the cells.

The efficiency of this method is much greater than the stand-alone classical N-body method and is typically around \( \sim O(N \log N) \). [Hernquist and Katz, 1989, Aarseth, 2003]. This is a fantastic speed up by itself, but with today’s computing resources, it is even more amazing! Figure 2.1 illustrates this point graphically for \( N < 100 \) particles.

### 2.2 Model Galaxies

Only two parts of the total galaxy structure need be considered for initial tests: the stellar disk and the dark matter halo. The code used for the theoretical N-body calculations, Gadget2, is capable of including additional particle types and it is certainly a small task to create extra initial conditions for a set of bulge particles, but it is not obvious whether
Figure 2.1: This graph shows the two functions $x^2$, (red), and $x \log x$, (green). Note the difference in scale for even a small number of particles.
the additional types would significantly affect the outcome during initial testing.

Of the two parts modeled, there is no question about the inclusion of a stellar disk; stellar disks are visible in every galaxy. However, the presence of a dark matter halo around galaxies was only discovered about thirty years ago and because of its invisible nature, very little is known about its structure. There are a number of experiments trying to determine the structure of the Galactic dark matter halo by observing the motion of objects inside it, but far away from the disk, [Battaglia et al., 2005].

The “smoking gun” for the presence of large amounts of unseen matter in galactic mass distributions came when observations of rotational curves failed to match theoretical predictions. Specifically, if the disks of galaxies are approximated as exponential disks and the circular velocity is plotted versus radius, the rotation curve is seen to drop after some small distance. Observed rotation curves, on the other hand, fail to drop off and flatten out at radii much greater than the edge of the visible matter. Fig. 2.2 shows the measured and predicted rotation curves for NGC-6503.

2.2.1 Exponential Disks

For this work, a disk model with an exponentially decreasing surface density is used as described in section 2.6 of [Binney and Tremaine, 1987].

The surface density for an exponential disk is,

\[
\Sigma(R) = \Sigma_0 e^{-R/R_d} = \frac{M_d}{2\pi R_d^2} e^{-R/R_d} \tag{2.4}
\]

where \( R_d \) is the scale length of the disk, \( M_d \) is the total mass of the disk, and \( \Sigma_0 = \frac{M_d}{2\pi R_d^2} \)

The gravitational potential is found by solving Poisson’s equation and for \( z = 0 \),

\[
\Phi(R, 0) = -\pi G \Sigma_0 R \left[ I_0 \left( \frac{R}{2R_d} \right) K_1 \left( \frac{R}{2R_d} \right) + I_1 \left( \frac{R}{2R_d} \right) K_0 \left( \frac{R}{2R_d} \right) \right] \tag{2.5}
\]

where \( I \) and \( K \) are modified regular cylindrical Bessel functions of the first and second kind.

There are two additional physical quantities needed to fully realize this disk: the mass

\footnote{This is not the easiest task. See [Binney and Tremaine, 1987] for a detailed discussion. In short, even for a distribution as simple as this, solving Poisson’s equation is difficult and requires judicious use of Bessel functions.}
Figure 2.2: Measured and predicted rotation curves for NGC 6503. Note the flattening of the observed curve versus the predicted values.
distribution and the rotation curve. The mass distribution can be found by integrating the surface density over the area of the disk

\[
M_d(R) = 2\pi \int \Sigma \cdot Rdr = 2\pi \Sigma_0 R_d^2 \left[ 1 - e^{-R/R_d} \left( 1 + \frac{R}{R_d} \right) \right]
\] (2.6)

The total mass of the disk \( M_d \) depends on the total mass of the dark matter halo, see section 2.2.3. The circular velocity comes from the partial derivative of the potential,

\[
v_c(R) = \sqrt{R \frac{\partial \Phi}{\partial R}} = \sqrt{4\pi G \Sigma_0 R^2 4 R_d \left[ I_0 \left( \frac{R}{2 R_d} \right) K_0 \left( \frac{R}{2 R_d} \right) + I_1 \left( \frac{R}{2 R_d} \right) K_1 \left( \frac{R}{2 R_d} \right) \right]}
\] (2.7)

Finally, it is necessary to give some value for the disk scale length \( R_d \), but this is also dependent on the nature of the dark matter halo.

### 2.2.2 Dark Matter Halos

The difficulty of determining the structure of an invisible dark matter halo has already been mentioned. Historically, these have been modeled using isothermal spheres, as in [Hernquist, 1993], but there are a number of observations that disagree with this type of halo. Indeed, very recent observations disagree with it completely, [Battaglia et al., 2005], and favor the NFW profile from [Navarro et al., 1996].

The NFW profile assumes a virialized halo with density structure,

\[
\rho(r) = \frac{\delta_c \rho_{\text{crit}}}{(r/r_s)(1 + r/r_s)^2}
\] (2.8)

where \( \rho_{\text{crit}} \) and \( r_s \) are

\[
\rho_{\text{crit}} = \frac{3H^2}{8\pi G}
\] (2.9)

\[
r_s = r_{200}/c
\] (2.10)

The critical density is the density of the dark matter before the halo formed, [Springel and White, 1999], and \( \delta_c \) represents the characteristic overdensity. This quantity is directly
related to the concentration, \( c = r_{200}/r_s \) by

\[
\delta_c = \frac{200}{3} \left( \frac{c^3}{\ln(1 + c) - c/(1 + c)} \right)
\]

(2.11)

and the criteria for \( \delta_c \) and \( c \) is a restriction that the density within \( r_{200} \) is exactly two hundred times the background, or critical, density. \( r_{200} \) represents the total mass of the halo

\[
M_{200} = 200\rho_{\text{crit}}(4\pi/3)r_{200}^3
\]

(2.12)

The mass as a function of radius is, [Mo et al., 1998]

\[
M(r) = 4\pi \int \rho(r) \cdot r^2 dr = 4\pi \rho_{\text{crit}}\delta_c r_s^3 \left[ \frac{1}{1 + cx} - 1 + \ln(1 + cx) \right]
\]

(2.13)

The circular velocity \( v_{200} \) at radius \( r_{200} \) with interior mass \( M_{200} \) is, [Springel and White, 1999]

\[
v_{200} = \sqrt{\frac{GM_{200}}{r_{200}}}
\]

(2.14)

and the circular velocity at a given radius for a spherical mass distribution is, [Binney and Tremaine, 1987]

\[
v_c(r) = \sqrt{\frac{GM(r)}{r}}
\]

(2.15)

The quantities \( v_{200}, r_{200}, \) and \( M_{200} \) are the virial quantities that ultimately set the nature of the halo. For a given \( v_{200} \), [Springel and White, 1999],

\[
M_{200} = \frac{v_{200}^3}{10GH}
\]

(2.16)

\[
r_{200} = \frac{v_{200}}{10H}
\]

(2.17)

A discussion of the Hubble constant has been avoided. For the purposes of this work, the Hubble constant will always be taken as 71.4 km s\(^{-1}\) Mpc\(^{-1}\).

### 2.2.3 Disk in a Halo?

To simulate galaxy collisions, a disk and halo must be combined. The process used to create initial conditions in this work is based on the method originally described by [Hernquist,
and adapted for the NFW profile and Gadget2 by [Springel and White, 1999].

[Springel and White, 1999] uses the analytical work of [Mo et al., 1998] to combine the disk and halo. Four assumptions are made in this model about the nature of the disk. All disks are approximated as exponential disks, just as above. Also, observed disks are assumed to be dynamically stable against bar formation. Finally, the last two can be combined in a statement of the specific angular momentum,

\[
\frac{J_d}{M_d} = \frac{j_d J_{DM}}{m_d M_{DM}} \tag{2.18}
\]

where \( j_d \) and \( m_d \) are fractions of the total. [Mo et al., 1998] also specify the total angular momentum of the halo in terms of a spin parameter

\[
\lambda = \frac{J_{DM}|E|^{1/2}}{GM_{DM}^{5/2}} \tag{2.19}
\]

and, by assuming the particles are all in circular orbits,

\[
E = -\frac{\pi r_0^2}{2} \cdot \left[ \frac{1}{2} - \frac{1}{2(1 + c)^2} - \frac{\ln(1 + c)}{1 + c} \right] = -\frac{G M_{DM}^2}{2 r_{200}} \cdot f_c \tag{2.20}
\]

for

\[
f_c = \frac{c}{2} \left[ 1 - \frac{1}{(1 + c)^2} - \frac{2 \ln(1 + c) / (1 + c)}{(c/(1 + c) - \ln(1 + c))^2} \right] \tag{2.21}
\]

Using the above description of the energy, the spin parameter, and the angular momentum,

\[
J_d = M_d R_d v_{200} \int_0^{r_{200}/R_d} e^{-u^2 v_c(R_d u)} du \tag{2.22}
\]

it is possible to write an equation for the disk scale length, which was mentioned earlier as being dependent on the particulars of the dark matter halo,

\[
R_d = \frac{1}{\sqrt{2}} \left( \frac{j_d}{m_d} \right) \lambda r_{200} f_c^{-1/2} f_R(\lambda, c, m_d, j_d) \tag{2.23}
\]

for

\[
f_R(\lambda, c, m_d, j_d) = 2 \left[ \int_0^\infty e^{-u^2 v_c(R_d u)} du \right]^{-1} \tag{2.24}
\]

1993], and adapted for the NFW profile and Gadget2 by [Springel and White, 1999].
The circular velocity of the system is

\[ v_c(r) = \sqrt{v_{c,d}(r)^2 + v_{c,DM}(r)^2} \]  

(2.25)

and to find the circular velocity of the dark matter halo, it is first necessary to consider the contraction of the inner portion of the halo due to the addition of the disk. This, then, gives a new condition on the total mass of the system,

\[ M_f(r) = M_d(r) + M_{DM}(r_i)(1 - m_d) \]  

(2.26)

such that

\[ v_{c,DM} = \sqrt{G(M_f(r) - M_d(r))/r} \]  

(2.27)

[Mo et al., 1998] make a special point that solving for \( R_d \) numerically requires iterating over a number of the above equations until the value of \( R_d \) converges. In possibly the best form of scientific altruism, they take care of the work and offer a fitting formula for \( f_R \) that removes the need for iteration,

\[ f_R \sim \left( \frac{(\frac{j_d}{m_d})\lambda}{0.1} \right)^{-0.06+2.71m_d+0.0047/[(\frac{j_d}{m_d})\lambda]} \cdot \]  

(2.28)

\[ \cdot (1 - 3m_d + 5.2m_d^2) \cdot (1 - 0.019c + 0.00025c^2 + 0.52/c) \]

With the above formulation by [Mo et al., 1998, Springel and White, 1999, Hernquist, 1993], it is possible to model a galaxy collision.

### 2.2.4 Initial Conditions

The initial conditions are created by using the formulation for a disk/halo pair and the Gadget2 code, written by Dr. Volker Springel, is used to realize the collision.

**Particle Number, Mass, Position and Velocity**

The initial conditions for the collision require first the total number of particles and second the position, velocity, and mass of each particle. In order to determine the mass of each
particle, first the total number of desired particles is chosen and then the virial velocity $v_{200}$ is picked. The total mass of the system is given by equation 2.17.

There are several other quantities that must be ‘picked’ at this stage. The concentration, $c$, the disk mass and angular momentum fractions, and the spin parameter $\lambda$. These choices are made just as in [Springel and White, 1999] and, also as stated there, $\lambda \geq m_d$. After these are set, it is possible to calculate the disk scale length. Calculating this is much easier if the fitting formula for $f_R$ is used, even though there is some inherent error involved.

Particles positions are randomly initialized by binning the mass in rings or shells, depending on the particle type, and counting the number of particles in the bin. To keep particles from stacking on top of each other, each particle position is smoothed by subtracting a uniform random deviate less than the width of the bin. At some radius, the total mass of the disk is allocated and at some even greater radius, the dark matter halo follows suit.

Particle velocities are initialized by assuming that each particle is moving with a speed exactly equal to the local circular velocity. The direction of the velocity is randomly chosen for the halo particles, but the disk particles all travel in the same direction. This is an inaccurate description of the velocities because real galaxies exhibit some degree of dispersion. Velocity dispersions are neglected altogether in the hopes that the aforementioned smoothing of the particle positions and low particle number per simulation avoid any possible particle encounters, at least initially.

Also as per [Springel and White, 1999], the centers of the galaxies are placed apart by twice the virial radius and set on a parabolic orbit.

**Gadget2**

Gadget2 is a full N-body and hydrodynamics suite that utilizes a tree algorithm, mentioned above, for the N-body routine and Smoothed Particle Hydrodynamics for gas modeling. The latter is not needed in this work because the disk and halo particles are not gas, but it will be needed in the future. There are additional unused particle types, like the bulge type alluded to earlier for bulge particles.
The greatest challenge of working with Gadget2 is that the format of the initial conditions file is binary and not the most easily accessible. However, Dr. Springel provides a file for reading these snapshots, read_snapshot.c, with the Gadget2 source code that is easily translated to a write routine.

Starscream

The Starscream code developed for this thesis is completely compatible with Gadget2 and performs all of the above initializations and writes to the binary format. The code is written in a combination of C and FORTRAN 90. Random numbers are provided by the implementation of the mt19937 algorithm provided in the GNU Scientific Library.

Starscream is a work in progress. As an example, routines exist in Starscream to calculate the velocity dispersions of the galaxies using the method outlined in [Hernquist, 1993], but the accuracy of these calculations is still in question and debugging is required. Furthermore, the mass binning system discussed above is overtly naive and it would be much nicer to pick particle positions from distribution functions.
Chapter 3

Feed-Forward Back-Propagating Artificial Neural Networks

Artificial Neural Networks are the stereotypical form of Artificial Intelligence. When a layman refers to AI that will one day take over the world, they are referring to neural networks that can recognize images, decipher human handwriting, decide whether or not to award a loan, filter through all the Russian spam in Thunderbird, or travel back in time to hunt Sarah Conner, (see Fig. 3.1.a.).

Except for the latter, all are present day applications of neural networks and these wonderful tools are appearing in many different walks of life. While using a neural network to look for patterns in theoretical models may be relatively new in astrophysics, astronomers have already adopted these tools for a number of applications. The works of [Odewahn et al., 1996, Daigle et al., 2003, Sarro et al., 2006] are just a few cases of precedence.

This chapter will describe the school of thought behind ANNs and introduce Megatron, the network written for this work. The theoretical description that follows is based mostly on [Haykin, 1999] and the practical examples of Dr. John Bullinaria.

3.1 The Artificial Neuron

Artificial neural networks are based on the human brain. Early computer scientists realized that the human brain works completely different than a normal computer and looked for a
Figure 3.1: (a) The Terminator is an example of a fictional artificial intelligence that travels back in time to change humanity’s future. (b) Mozilla Thunderbird uses Self Organizing Maps, an advanced form of neural network, to filter junk email.

way to simulate the behavior of the brain. The “processing unit” of the brain is the neuron, a special purpose cell that communicates via electrical impulses. Neurons take an input, process it, and then fire a new electrical impulse. Furthermore, neurons exhibit a high degree of parallelism between each other. The human brain has roughly $10^{11}$ neurons, each with approximately $10^4$ connections to other neurons. The strength or weight between two real neurons is a chemical connection that grows stronger for increased activity between two neurons.

The implications of the above information may not be entirely obvious. The real point is that neurons have very nonlinear behavior and that a change in one neuron can affect a huge number of its neighbors. This is in stark contrast to traditional computing where each transistor takes two inputs and fires to the next transistor in line, without talking to the two transistors that turned it on or off.

Artificial, (or virtual), neurons behave in much the same way as real neurons. A value is input into the neuron and the output is computed by a non-linear activation function. For more than a single input neuron, weights are applied to adjust the value and a summation is performed over all the inputs. Let $x$ be the value of an input, $\omega_i$ the weight, and $y$ the output, then,

$$y = f \left( \sum_i^N \omega_i x_i \right)$$

(3.1)

Fig. 3.2.a shows an artificial neuron and Fig. 3.2.b shows a small network. In Fig. 3.2.b, the reader will note the addition of a second, hidden layer, between the inputs and output.
neuron. There are many reasons to include a hidden layer, but for the purposes of this work it is enough to say that inclusion of a hidden layer allows the network to universally approximate features in an image.\footnote{This is actually a fundamental theorem of neural networks; The Universal Approximation Theorem was proven by Auer, Burgsteiner, and Maas.}

### 3.1.1 Activation Functions: sigmoid vs. tanh

The choice of activation function is important. Typical choices are members of the sigmoid family, i.e. - the sigmoid function and those functions like it. Fig. 3.3 shows a graph of the sigmoid function and a popular alternative; the hyperbolic tangent function. The sigmoid function is,

$$ y = \frac{1}{1 + e^{-x}} \quad (3.2) $$

and the hyperbolic tangent is

$$ y = \tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \quad (3.3) $$

There is a very important difference between the two functions that should not be taken lightly. Notice that the output of the sigmoid function is bounded between 0 and 1, but the output of the hyperbolic tangent is between -1 and 1. This can have very tangible

Figure 3.2: (a) An artificial neuron. (b) A small artificial neural network with 1 hidden layer. Note the high interconnectivity between neurons.
Figure 3.3: This graph shows the sigmoid function, (red), and \( \tanh(x) \), (black). Note the difference in output values.

3.2 Error Back-Propagation

The utility of the neural network comes in its ability to learn and model a certain dataset. This is achieved by changing the weights between nodes to produce a desired output, but doing this efficiently and correctly can take some work.

The Error Back-Propagation algorithm is a common choice for changing the weights. The idea is this: compare the output value of the network to a desired output value, measure the error, and update the weights via gradient descent to minimize the error. This is often referred to as learning with a teacher.
For the target output value \( t \) and network output \( o \), the sum-squared error is [Haykin, 1999],

\[
E = \frac{1}{2} \sum_{j=0}^{N} (e_j)^2
\]  

(3.4)

for

\[
e_j = t_j - o_j
\]  

(3.5)

To minimize this error function, find its derivative with respect to the weights,

\[
\frac{\partial E}{\partial \omega_{ji}} = -e_j \frac{\partial f_j(x)}{\partial x} f_i(x)
\]  

(3.6)

where \( \omega_{ji} \) are the weights between neurons. It is desirable to use a learning rate, \( \eta \), to keep from overshooting the target, and

\[
\Delta \omega_{ji} = \eta e_j \frac{\partial f_j(x)}{\partial x} f_i(x) = \eta \delta_j f_i x
\]  

(3.7)

This direct approach only works for the output layer. For the hidden layer, \( \delta_j \) becomes

\[
\delta_j = \frac{\partial f_j(x)}{\partial x} \sum_k \delta_k \omega_{kj}
\]  

(3.8)

where \( j \) now represents a hidden node and \( k \) goes over outputs. This algorithm is sufficient to iteratively train the network.

**Biasing**

The above discussion rolls over an important part of neural networks which is often left untouched. Because the network needs to deal with translations in the dataset, it is important to include a bias neuron. It is customary to leave this element out of diagrams because it assumes a constant value of 1.0 and its contribution is merely the value of its weights.
3.3 Megatron

The artificial neural network written for this work is *Megatron*. *Megatron* is a simple feed-forward, back-propagating neural network that functions according to the system described above. *Megatron* is also capable of dynamically scaling the input layer of the network and globally stores weight information to increase performance. There are three possible operating modes with *Megatron*: a pure training mode, a mode to add information to the training set, and a general operation mode.

3.3.1 Test Case: To Catch a Terrorist

To test *Megatron*, a simple exercise was designed with national security in mind. The training sample included images of Osama bin Laden with target output values of 1.0, and an image each of President George W. Bush, Karl Rove, and Hillary Clinton all three with target output values of 0.0. Each image was edited to have a uniform background and set to grayscale. The images were 200x185 PNG files rastered into an array of length 37000 and the number of hidden nodes was 150. The code was trained to a mean error of 0.01.

Images of bearded and unbearded members of the UTK Physics department were assembled as test images in the same fashion as the training images. Two extra images were added to the test set: an image each of Osama bin Laden and President George W. Bush. These extra images are for control and were not part of the training set. When these images were rastered through the network, (now set to the general operation mode), the network reported very low output values for each member of the UTK Physics department and President George W. Bush, between 0 and 0.2, and correctly identified the image of Osama bin Laden with an output value of 0.986.

In a hand waving way, one could say that the test here is to see if bearded members of the department “look like terrorists.” However, this is actually a test to see if bearded members of the department look more like Osama bin Laden than members of the U.S. Government. This is a very useful test when the hand waving argument is taken seriously and multiple targets are trained instead of a single, well-known example.

Fig. 3.4 shows the mean error and the output values for this test case. Note the relatively high output value for the control photo of Osama bin Laden.
Figure 3.4: (a) The mean output error of the network as a function of training epoch. (b) Output values of the network in the test case.
Chapter 4

Genetic Algorithms

Genetic algorithms are a second form of Artificial Intelligence modeled after something natural. Where the Artificial Neural Network is an abstraction of the human brain, the Genetic Algorithm is an abstraction of Evolution.

The Genetic Algorithm is a heuristic search technique, [Charbonneau, 1998], that is particularly suited for \textit{global optimization}. There are four things that set genetic algorithms apart from traditional calculus, enumerative, or random techniques, [Goldberg, 1989]. Genetic algorithms...

1.) do not work with the problem parameters directly.
2.) use populations of potential solutions.
3.) make decisions by fitness of solution, not projection.
4.) are probabilistic.

This chapter will provide a short description of the pieces inside a genetic algorithm and discuss the \textit{OptimusPrime} code developed for this work.

4.1 Parts of a Genetic Algorithm

There are many parts of a genetic algorithm, but the story starts with the parameters of a given function. Consider the case of a 2D Gaussian, pictured in Fig. 4.1 and given by,

\[ z = e^{-(x^2+y^2)} \]  

(4.1)
Any value of $z$ on the surface is given by an $x, y$ pair. Take this pair as an individual with two genes, one $x_1$ and one $y_1$, that make up its chromosome. A second individual would have a chromosome $x_2, y_2$, a third $x_3, y_3$, etc. The total group of individuals defines the population.

In the case of the 2D Gaussian, the maximum possible value is $z = 1$ for $x = y = 0$. This value represents the maximum fitness of the function and any $x, y$ pair that yields $z \sim 0$ represents a very poor fitness. Thus, each member of the population has a certain fitness depending on their genetic material and are fit, more or less, to the environment.

In true evolutionary fashion, it is possible for two chromosomes to walk into a bar. That is, it is possible for individuals to meet and produce offspring. This is called *crossing over* in the jargon and, for our example, this would represent a switch in the $y$ value of the individual.\footnote{This description is perhaps overly simplistic. See [Klug and Cummings, 2000].} Additionally, it is possible for this new member of the population to have a mutation such that $x$ or $y$ becomes $x'$ or $y'$ or both. These new members also have some
fitness to the environment, determined by the $x, y$ values that they inherited from their parents.

This process continues for either a set number of generations or until a member of the population reaches a certain degree of fitness.

### 4.1.1 More on Crossover and Mutation

Crossover and mutation are arguably the most important parts of a genetic algorithm because they determine the nature of the next generation. There are many different types of both and a few examples are presented here.

Define a 1D vector to be

$$\vec{x} = [0, 1, 0, 1, 0, 1, 1, 1] \quad (4.2)$$

and a second to be

$$\vec{y} = [1, 1, 1, 0, 1, 0, 1, 0] \quad (4.3)$$

The simplest cross between these two vectors happens when they are split at a single point, usually the middle, and recombined. This is called one point crossover and yields two offspring,

$$\vec{a} = [0, 1, 0, 1, 1, 0, 1, 0] \quad (4.4)$$

and

$$\vec{b} = [1, 1, 1, 0, 1, 1, 1, 1] \quad (4.5)$$

It is possible to perform two point crossover or even a multi-point crossover by performing the cut at two or many points on the chromosome. The benefit to higher crossover terms is that each offspring can have multiple parts of the parent. (The utility of this is obvious if the above vectors are for an 8D Gaussian, for example.)

Mutation can occur randomly on offspring in a population. Example of mutations on $\vec{a}$ are,

$$\vec{a} = [1, 1, 0, 1, 1, 0, 1, 0] \quad (4.6)$$

or

$$\vec{a} = [1.1, 1, -0.1, 1, 0, 1, 0.8] \quad (4.7)$$
Mutation is very important and is the sole factor that keeps an entire population from centering on a local minima or maxima.

**Overpopulation?**

Population control in genetic algorithms is performed by a number of methods including worst replacement and full generational replacement. In the first case, the members with the worst fitness are replaced by the offspring of each generation. In the latter, the entire population is replaced at the end of a generation.

### 4.1.2 Maximizing Functions

Any standard calculus-based routine can find the the maximum point of a Gaussian. What about more complicated functions? The benefit of genetic algorithms is that they maximize globally. Individual members of the population will cross with very different members of the population based on fitness as opposed to one point being iteratively pushed up a hill.

Consider the function in Fig. 4.2 given by

\[ z = \frac{\sin(x)}{x} + \frac{\sin(y)}{y} \] (4.8)

An interative hill climbing technique will would maximize this problem locally and could possibly get stopped at any one of the many hills. OptimusPrime maximizes this function correctly to within 1% in 0.1 seconds over the range of [-10,0,10.0] on an AMD Turion64x2 laptop with 1GB of RAM.

### 4.2 OptimusPrime

This work originally planned to use Pikaia, written by [Charbonneau, 1998], but the plans were changed in an effort to keep as much of the code in the same language as the scientific engine, Gadget2, as possible. Therefore, the C programming language won and Pikaia was used as the basis for a new genetic algorithm, OptimusPrime.

At the most basic level, OptimusPrime is very similar to Pikaia. However, OptimusPrime performs a number of things differently than Pikaia to tailor it a little more to
Figure 4.2: This is a graph of $\frac{\sin(x)}{x} + \frac{\sin(y)}{y}$
programs, not functions, with many parameters. Some of the important points about OptimusPrime include real-valued genes, (i.e. - the gene values are the values used in the code, within a multiplicative constant), and the possibility to use multi-point crossover. However, OptimusPrime makes a small sacrifice in mutation; advanced mutation schemes are not implemented and, instead, the code merely “bumps” values by adding a small amount.

OptimusPrime uses a large amount of random numbers. If available, the Intel C compiler speeds up random number generation, and therefore the execution time, by a factor of two.
Chapter 5

Results and Conclusions

This chapter will discuss results, a to-do list for future work, and the possibility of using this method in other codes.

5.1 Results: Transforming Galaxy Collisions

The previous chapters described three different codes with three very different purposes. These codes are now wired together with a small, easily changeable wrapper function written in C. The genetic algorithm is used to generate initial population members whose chromosomes define the parameters for Gadget2. After a population is defined, each member is checked for fitness, which is calculated by passing an output image from Gadget2 to the artificial neural network trained on images similar to those in Chapter 1. The fitness values are returned to the genetic algorithm and the system repeats the process for a number of generations. Therefore, the galaxy collision code Gadget2 and Megatron become elements of the fitness function of the genetic algorithm and only a single number, the output error of the neural network, is returned as a measurement of fitness.

The work to date has produced very few results. However, the initial conditions code Starscream has generated some interesting interacting systems and the neural network Megatron is being trained to better recognize colliding galaxies.
5.1.1 Galaxies

The reader will recall that there are several parameters that must be specified to create the initial conditions for a galaxy collision: the spin parameter $\lambda$, the disk mass fraction $m_d$, the disk angular momentum fraction $j_d$, the virialized circular velocity $v_{200}$, and the halo concentration $c$. Fig. 5.1 shows the results of a test case of the initial conditions generator Starscream and Gadget2 for $\lambda = m_d = j_d = 0.025$, $v_{200} = 160 km/s$ and $c = 15$. The galaxies in this example are of equal mass and are separated by a distance of $2r_{200}$ on parabolic orbits. Each galaxy contains 10,000 disk particles and 20,000 halo particles, for a total of 60,000 particles.

A second test of the code was performed with an identical set of initial conditions, except with one disk in the xz plane. The results of this test are shown in Fig. 5.2 and it is immediately apparent that this collision creates tidal tails in different directions than the first test case.

5.1.2 Training

The greatest challenge for this work lies in the neural network: training is a notoriously difficult task and has hindered the investigation. The terrorism test case was successfully trained because of the diversity and number of members in the training set, but work remains to properly define a training set for the galaxy collision problem. Fig. 5.3 shows the three interacting systems pictured in Chapter 1, but processed to remove large stars, color, and optical noise. A minimally processed image from the second galaxy collision test is also pictured for comparison.$^1$

It is the author’s opinion that three images are insufficient for a positive training set and that these images, taken from the public Digitized Sky Survey, are of poor quality. Therefore, it may be beneficial to observe these and other collisional systems to obtain a diverse, standardized and high resolution training set of many images.

Because of the time required to run one small simulation of Gadget is on the order of one hour for $\sim 100,000$ particles, it will take some time for this code to generate significant results and much longer to check those results for errors. Even a small population size of

---

$^1$The reader should note the similarity of this to Fig. 5.3.b.
Figure 5.1: This is a composite image of a galaxy collision with both galactic disks in the $xy$ plane. Snapshots are taken at $t = 0$, $0.15t_h$, $0.19t_h$, $0.21t_h$, $0.3t_h$ and $0.33t_h$, where $t_h$ is the Hubble time.
Figure 5.2: This is a composite image of a galaxy collision with one galactic disk in the xy plane and a second in the xz plane. Snapshots are taken at t = 0, 0.15t_h, 0.19t_h, 0.205t_h, 0.3t_h and 0.33t_h, where t_h is the Hubble time.
Figure 5.3: The first three images are part of the network training set, the last image is Gadget2 output using the initial conditions of galaxy collision 2. (a) NGC 4038/9 “The Antennae,” 15’ x 15’, north up (b) NGC 4676 “The Mice,” 5’ x 5’, north up (c) NGC 7252, 7’ x 7’, north up. (d) Image output from Gadget2, processed in Starsplatter 2.0, t ∼ 3.0t_h

fifty members running for fifty generations will take a few days.

5.2 Conclusions: The Future

A lot remains for this work to become useful, but the pieces have been put into place for, potentially, a very powerful tool. There are a number of possible improvements over the current implementations of Starscream, Megatron, and OptimusPrime. Some of those possibilities are listed here along with a brief description of the potential gain.

5.2.1 Galaxy Collisions: Starscream

To provide a realistic analysis of the galaxy collision parameter space, it will be necessary to increase the number of particles in the simulation to millions of particles. The first and foremost problem, in this case, is dispersion. Not only will large numbers of particles create a system that is much more realistic, but they will require a system that is much more accurate. Any comparison of the results to actual observations will also require realistic
velocity dispersions, if not some other adjustments. It is also possible that the code will run much faster for systems with realistic dispersion because the systems will be much more stable.

Future simulations will also require gas and bulge particles. The process of adding bulge particles is outlined in [Springel and White, 1999] and [Hernquist, 1993], but neither paper gives a discussion of adding gas particles. As previously mentioned, Gadget2 is fully capable of running SPH calculations and the addition of gas particles will create much more realistic collisions, both physically and visually.

Given the age of the papers used as reference material, it would be at least interesting to check for updates in the science behind galaxy collisions. A good example of this would be observational or theoretical updates to the dark matter profile as almost happened in [Battaglia et al., 2005] and as was done in [Eke et al., 2001]. (The details are left to the reader.)

Finally, Starscream should “pick a language” and be fully implemented in either C or FORTRAN 90, but not both. This will allow for standardization of the algorithms used for integrations and random number generation; the current version uses both C and FORTRAN codes for those. This improvement would be mostly aesthetic, but speedup could be gained from generating random numbers in the same language. If the code is to be deployed in a very large, massively parallel attempt to use the AI suite, its functionality with vectorizing compilers like the Intel compilers should be checked thoroughly.

5.2.2 Pattern Recognition: Megatron

One of the biggest problems in training an Artificial Neural Network is determining the size of the network. Too many hidden nodes can result in overly long training times and too few hidden nodes can result in a failure to properly train at all. A system that dynamically determined the size of the network could dramatically improve the performance of the network over very long run times even with the overhead of checking outputs and removing or adding nodes.

Another nice feature would be a built-in wavelet analysis tool for removing unneeded parts of the training images and, therefore, decreasing the training time. Alternatively, the
activation function of Megatron could be changed to a wavelet basis, or “mother,” function, which would create a type of wavelet neural network that could do wavelet analysis on the fly.

Megatron was based on a very common “textbook” design and is therefore very simple. This program uses the most basic measurements for the error from the target as well as an aging method of back propagating the error. While the code works, alternatives to sum-squared error and the standard back-propagation algorithm exist that could, after an exhaustive profiling, possibly improve the performance. (It would even be possible to use runtime switches for switching between algorithms and “maximize” the output of the network with OptimusPrime!)

Alas, there are not too many improvements to be made on FFBP-ANNs; the technology is a little old.

5.2.3 Parameter Sweeping: OptimusPrime

Most of the improvements that could be made to OptimusPrime are code optimizations that are probably fixed at compile time, especially when the Intel C compiler is used. However, new implementations of crossover and mutation are constantly developed and the survival of this code as a competent genetic algorithm will be dependent, (not unlike the populations it creates), on adapting these new methods into the code.

5.3 Other Codes

Testing this AI suite with other scientific engines will be key to establishing these tools as both useful and worthwhile. In the short-term, it is enough to make improvements to Starscream and map the GC parameter space. Instead of Gadget2, it should also be possible to use N-body codes such as Dr. Joshua Barnes’ Zeno code or Dr. Sverre Aarseth’s NBODY series of codes.

The charge of the author was to create a code suite that could replace the need for manual parameter sweeping in the galaxy collision problem while keeping the suite general enough to apply it to other problems. To date, this is the only large problem tested with the suite, but one immediately notices that other possibilities exist.
One such problem is thermonuclear burning in astrophysical environments. Nuclear reaction networks can be profiled and the effective rates optimized with this suite. In this case, the values of key rates would be changed by the genetic algorithm and physical quantities, such as isotopic mass fractions and energy production, would be used as inputs in the neural network. *Starscream* would be replaced completely with a simple network solver or an IO routine to edit the initial conditions file of a network solver that would evolve the network over time.
Bibliography
Bibliography


Vita

Jay Billings was born on June 14, 1984 and raised in Marion, Va. In the spring of 2005, he completed his undergraduate career at Virginia Tech with a B.S. in Physics and minors in Astronomy and Mathematics. He matriculated at the University of Tennessee the following fall and graduated with the Master of Science degree in Theoretical Astrophysics in the spring of 2008. His sole desire in life is to enlist in the United States Army as a pastry chef.