Modeling Trabecular Microstructure Evolution via Genetic Algorithm

by

Samuel W. L. Shames

Submitted to the
Department of Materials Science and Engineering
in Partial Fulfillment of the Requirements for the Degree of

Bachelor of Science

at the

Massachusetts Institute of Technology

June 2013

Signature of Author.....

Deparment of Materials Science and Engineering
May 3, 2013

Certified by.....

W. Craig Carter
Professor of Materials Science and Engineering
Thesis Supervisor

Accepted by.....

Jeffrey C. Grossman
Department of Materials Science and Engineering
Undergraduate Committee Chairman

ACKNOWLEDGEMENTS

This work represents the culmination of the last three years of my education. I am deeply indebted to everyone who helped me to reach this point, and there are a number of people without whom this thesis would not have been possible. I would like to thank a few key individuals for their help and support.

First, I would like to extend my sincerest thanks to my thesis advisor, Professor W. Craig Carter. Professor Carter has been a mentor and resource for me since I took 3.016 with him. I was first impressed by his bag of tricks in Mathematica and later by the range and scope of his knowledge of materials science, art, The Wire, and more. His willingness to help, along with his seemingly endless list of interesting problems was the perfect complement to my curiosity about materials science and Mathematica. I loved how he always challenged me and pushed me to create work that was outstanding and beautiful. As my advisor, Professor Carter suggested my thesis topic and has continually provided me with time and assistance as I struggled to conduct this research. For everything from the use of his computer cluster to his unparalleled patience with my unannounced visits to his office with questions, I am extremely grateful.

I would also like to thank Professor Lorna Gibson, in whose class, 3.032, I first learned about what a remarkable material bone is and about the ways in which it adapts itself to its environment. Hearing about the experiments with the guinea fowl was particularly memorable. After I had chosen a thesis topic, Professor Gibson was kind enough to meet with me to give me a series of papers all about the adaptability of trabecular bone. Her resources served as the basis for my literature review and proved incredibly valuable throughout the project. For both her excellent teaching and her enthusiastic willingness to help, I am grateful.

The last individual I would like to recognize is Professor Jeffrey C. Grossman. From the very first 3.012 lecture, I found Professor Grossman's charisma and passion for materials science inspiring, and it was during that class that I began to love the subject. Since finishing 3.012 and Professor Grossman's other class, 3.021, he has continued to inspire my education and has provided me with opportunities as a UROP and later as a TA that I didn't realize were possible. For nurturing my enthusiasm and passion for the subject, and for his mentorship, I offer my gratitude.

I also must thank all the other individuals who have supported me over the years and contributed to my growth as a learner. This includes my family and friends, as well as all the teachers I have had both at MIT and in high school. I must make offer an extra thanks to Mr. Steve Chinosi, the teacher who taught me to love learning. Without him and everyone else, I would not be in this position today. Thank you all for your continued support and for igniting my intellectual curiosity.

Modeling Trabecular Microstructure Evolution via Genetic Algorithm

by Samuel W. L. Shames

Modeling Trabecular Microstructure Evolution via Genetic Algorithm

by

Samuel W. L. Shames

Submitted to the Department of Materials Science and Engineering
on May 3, 2013 in Partial Fulfillment of the
Requirements for the Degree of Bachelor of Science in
Materials Science and Engineering

ABSTRACT

Connecting structure to properties, and optimizing properties by controlling structure is one of the fundamental goals of materials science and engineering. No where is this connection more apparent than with biomaterials, whose unparalleled properties are the result of the evolution via cumulative selection of highly specialized structures. Beyond biomaterials, cumulative selection offers a generalizable model for materials optimization via accumulative of beneficial mutations in a material's genome that improve the properties for a given function. A genetic algorithm is one method for applying the principals of cumulative selection to material's optimization.

One of unique property that cumulative selection generated was the ability of trabecular bone to optimize and adjust its structure *in vivo* in response to changes in its loading conditions. This work presents a model for trabecular microstructure evolution using a genetic algorithm, the same mechanism through which that ability evolved. The algorithm begins by translating a trabecular genome into a developed structure. It then simulates the structure's response under an applied load and selects for the genome which translates into the best structure. The selected genome is then replicated and mutated. Simulations of microstructure evolution consist of iterating through this process across multiple generations. A series of simulations was conducted demonstrating the ability of the algorithm to improve trabecular architecture. The systems tended to converge to a uniform stress distribution, after which additional generations of evolution had no effect on performance. During the simulations it was found that the length of the computation was most sensitive to the number of offspring per generation. Although focused on trabecular microstructure, this work establishes the use of a genetic algorithm as a general tool for material's optimization.

Thesis Supervisor: W. Craig Carter
Title: Professor of Materials Science and Engineering

Contents

1 Introduction **12**

2 Modeling Trabecular Microstructure Evolution **17**

3 Genetic Algorithms and Cumulative Selection **21**
3.1 The Power of Cumulative Selection 21
3.2 How Cumulative Selection Works 23
3.3 Cumulative Selection and Genetic Algorithms 24
3.4 A Genetic Algorithm for Radiating Lines 26

4 Creating a Genetic Algorithm for Trabecular Microstructure Evolution **33**
4.1 Characterizing Trabecular Microstructure 34
4.2 Trabecular Architecture and Graph Theory 35
4.3 Generating and Developing Trabecular Microstructures 38
4.4 Trabecular Architecture Genome 43
4.5 Generating Trabecular Architecture 44
4.6 Trabecular Loading and Selection 49
4.7 Reproduction 52
4.8 Non-Dimensionalizing 53
4.9 Evolution Function 54

5 Simulating Trabecular Microstructure Evolution **58**
5.1 Varying the Genome 60
5.2 Varying the Points 60
5.3 Mapping Genomes onto Points 62
5.4 Varying the Applied Stress 63
5.5 Decrease in Standard Deviation 63
5.6 General Trends 65

6 Discussion **68**
6.1 Number of Generations 69
6.2 Number of Offspring 70
6.3 Mutation Rate 71

6.4 Rate Limiting Computation 74
6.5 Limitations of Genetic Algorithms 75
6.6 Comparison with Previous Work 77
6.7 Future Work 78
6.7.1 Larger Scale Trabecular Simulations 78
6.7.2 Changing the Trabecular Genome 80
6.7.3 Extension to 3D 81
6.7.4 3D Printing and Experimental Validation 81
6.8 Genetic Algorithms for Materials Optimization 82

7 Conclusion **84**

8 References **86**

List of Figures

1.1 Schematic of the steps of the evolution process. 13
1.2 Schematic of the different levels of a materiome and the way in which they combine together to create hierarchical structures across length scales to generate the macroscopic elements of a material's structure. Reproduced from Cranford et al. 2012. 14
1.3 (a) SEM micrographs of cellcular cancellous bone structure reproduced from Gibson 1985. (b) Trabecular microstructure generated via genetic algorithm. 15

3.1 An example image of the types of radiating line pattern the genetic algorithm can produce. 27
3.2 The optimal radiating line structure based on the selection criteria of the maximum standard deviation in line length. 29
3.3 The evolution of one set of outputs of radiating line structures for a high, medium, and low mutation rate. Shown are the first, 10th, 20th, 30th, 40th, and 50th structures that were generated by the algorithm. In the medium and high mutation rate cases, the optimal structure was found by the end of the 50 generations. 30
3.4 A plot of the average standard deviation vs. generation number for 10 runs each at a low, medium, and high mutation rate. The mean standard deviation increases at a faster rate with a higher mutation rate. However, at high frequencies mutations are more likely to cause a decrease in standard deviation. Lower mutation rates are less likely to lead to regression. 31

4.1 Even though the vertices of these two graphs are located at different points in space, the graphs are isomorphic because the edges between the vertices are the same for both graphs. Unlike these graphs, the location of the vertices is a defining feature of the trabecular architecture; changing the location changes the overall structure. 36
4.2 A complete graph is one of the most basic graphs, where every vertex is connected to every other vertex. A complete graph mapped onto a trabecular architecture would mean that every vertex connects to every other one, resulting in a number of struts equal to the sum from 1 to the number of veritices minus one. 37

4.3 A complete graph with 100 vertices where each vertex is a point on this 10 by 10 lattice. Once the vertices and edges have been mapped to a lattice, they now be defined in terms of their geometry. 39
4.4 A random image that is used to generate the set of vertices within a trabecular cell. The pixel values in this 250 by 250 grayscale image, random numbers between 0 and 1, are used to determine a probability distribution when randomly selecting pixels to serve as vertices for struts. A pixel with an intensity closer to 1 is more likely to be selected, resulting a random weighted pixel distribution function. 40
4.5 Each of the four colors represents the randomly selected points for a region of space from a random image like the one in figure 4.4. Each of the four regions has a different number of selected pixels. A trabecular architecture will have a gene for each region that determines the number of points that are selected from the random image. These points serve as the vertices from which struts can be created. 41
4.6 The top 10, 100, and 500, struts based on the results of the strut selector function. The struts were compared to a strut of length 1, with angle Pi/2, and of thickness 0.01. 42
4.7 The results of translating a trabecular genome onto a set of lattice points. The genome specifies the number of struts that are created, the prefered angle, thickness, and length of a strut, and the weights given to each of those three strut features. 43
4.8 Trabecular architecture created by the combination of 16 different subcells, like the ones in figure 4.7. For these two architectures, the genome is the same for all the sub cells; however, the genome for the figure on the left is different than for the figure on the right. 45
4.9 A translation for a trabecular genome mapped to a random set of lattice points. Each cell had the same gene that specified that 100 lattice points would be chosen at random. The genome also specified that 200 struts would be created in each cell, that the ideal strut had thickness 0.005, length 1, and angle of Pi/2. It also specified that the angle was 3 times as important as the length and thickness. 46
4.10 A visualization of a trabecular architecture made of 16 different genomes mapped onto 16 sets of lattice points. 47
4.11 Trabecular microstructure for 16 independent genomes. Each cell has a unique genome that determines the number of random lattice points that were selected, the number of struts that were created, and the features of those struts. 48
4.12 Visualization of the stresses on a strut in a cell in a trabecular system. Blue coloring indicates low stress, while red indicates a large stress 51
4.13 Visualization of the stresses in a two by two supercell. 51
4.14 Schematic of trabecular microstructure evolution via genetic algorithm. . . . 54

4.15 Evolution of a small test trabecular architecture with random loading conditions. The struts are colored according to their strain energy density, with red representing a strut with a large stored energy. 56
4.16 Stress standard deviation for the trabecular architecture from figure 4.14. . . 57

5.1 Trabecular Architectures for evolution for two different genomes mapped onto the same set of points under the same stress state at three stages of evolution. The figure shows how drastically the microstructure of the system can change in only a small number of generations and how the general tendency is for the number of struts with a large strain energy—bright red struts—to decrease. . 61
5.2 The black struts represent the initial average length, thickness, and orientation of each cell in the two systems. The red struts represent the average strut at the end of ten generations of evolution. 62
5.3 The average initial and final struts in each cell in the two systems shown in black and red respectively. 63
5.4 The initial and final architectures of two identical genomes mapped onto two different set of points both under the same stress state. 64
5.5 The average struts for a single genome mapped onto five different sets of points. The length and thickness of the average strut seem to be invariant to the set of points onto which they are mapped, but the average angle can vary. 65
5.6 Showing how the average features of a genome vary with the points onto which it is mapped. The average angle appears to be more sensitive to changes in mapping points than the thickness or length. 66
5.7 Graph of the standard deviation in strain energy divided by the generation zero standard deviation shows how the system improves with increasing number of generations. The larger blue dots are for the first genome, while the smaller red dots are for the second genome. 67

6.1 The graph of standard deviation for different numbers of generations shows the effect of convergence: once the system finds the minimum standard deviation additional generations of evolution have little effect on performance. The largest dots, in red, are for the system that under went 25 generations of evolution. Decreasing dot size, from blue to green, to brown, to black, corresponds to the different number of generations of evolution the different systems underwent. 69
6.2 The graph of Standard Deviation for the larger systems shows how the trend of convergence is even more pronounced. Again, the dot size is proportional to the number of generations of evolution the different systems underwent. . 70
6.3 Increasing the number of offspring makes the system's performance level more stable and reduces the variability in standard deviation with subsequent generations 71

6.4 For the larger system, the improvements in stability are even more pronounced; the trial with 10 offspring per generation had variability nine orders of magnitude lower than the trial with only two offspring, as shown on this log plot. 72
6.5 In the large scale system, increasing the mutation rate did not greatly affected variability, presumable because the number of mutations that produced a system with equivalent performance was large enough that one was always produced. This time, the point size scales with mutation rate, with the largest mutation rate corresponding to the set of red points and the smallest corresponding to the black. 73
6.6 The small scale system showed that increasing the mutation rate did have an effect on system stability. Again, the mutation rate scales with the point size—from red to blue, to green, to brown, and to black 74
6.7 The simulations with a larger mutation rate tended to have a larger variability in converged stress than those with a smaller mutation rate. 75
6.8 The plot shows in red the increase in length of computation with varying number of offspring per generation while the blue dots are for varying number of generations. The steeper slope of the red line indicates that time is more sensitive to the number of offspring. 76
6.9 Trabecular architecture for a 64 cell system. 79

List of Tables

3.1 Partial range of genes for radiating line genetic algorithm. 27

5.1 Table of parameters of trabecular evolution simulations 59

Chapter 1

Introduction

Whether in metals, polymers, or any other class of materials, materials scientists attempt to connect structure to properties and to optimize properties by tuning and controlling structure. Understanding the structure-function-properties-performance relationship and using it to design novel materials is a fundamental goal of materials science and engineering. There are many methods to optimize the structure for a given function and one that has proven particularly successful is the process of cumulative selection used during natural selection.

Evolution via natural selection has produced some of the most remarkable materials ever synthesized: DNA, spider silk, enzymes, and tissue. Biomaterials are created by translating instructions encoded in an organism's DNA. These materials evolved through mutations in the set of genes encoding them, and through the gradual accumulation of beneficial mutations over millions of years they acquired the remarkable properties they have today. Figure 1.1 shows a schematic of the process of evolution via cumulative selection to which a set of genes for a biomaterial is subject.

Part of the power of cumulative selection is how generalizable it is; the same mechanism is use across all scales of evolution, not just the design of biomaterials. In materials science,

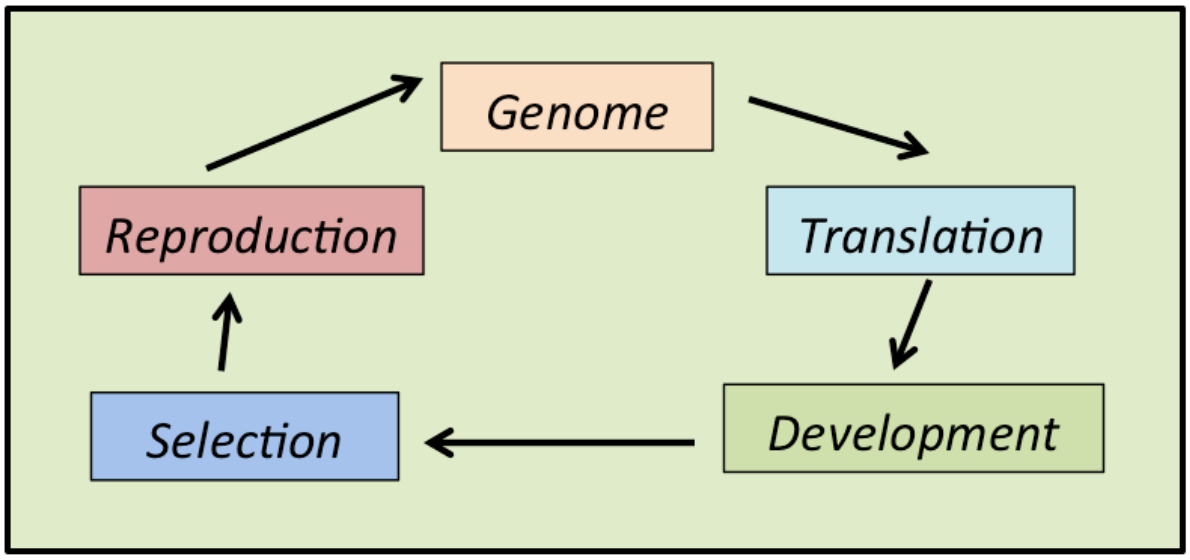

Figure 1.1: Schematic of the steps of the evolution process.

cumulative selection could also be used to probe the structure-function relationship for non-biological materials, like metals and polymers, and as a method for discovering optimal structures of these systems for a given application. As tool for materials scientists, cumulative selection represents a powerful method for materials development and optimization.

One special feature of biomaterials that cumulative selection generated is there unparalleled adaptability. Unlike inorganic materials, biomaterials have the unique ability to change their structure *in vivo* to adjust their properties in response to changes in their environment. Part of the way in which biomaterials accomplish this is through the development of hierachical structures, and by generating active sensing loops across the different length scales (Cranford et al. 2012). The creation of such feedback loops allows for multi-scale adaptation, where changes in the building blocks of the different scales create improved properties for better functionality. One way in which to characterize the hierarchical nature of biomaterials is through a materiome, which attempts to capture the material's structure-function relationships across all length scales. Figure 1.2 shows a schematic of the different levels of

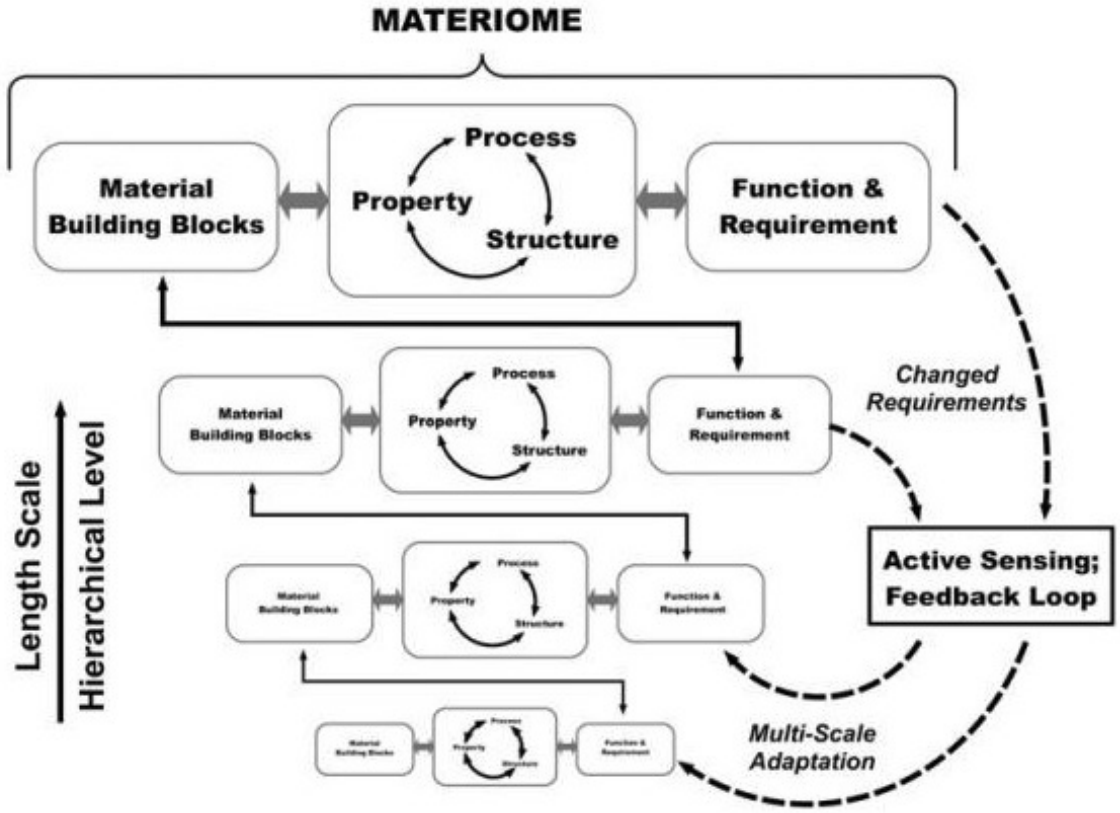

Figure 1.2: Schematic of the different levels of a materiome and the way in which they combine together to create hierarchical structures across length scales to generate the macroscopic elements of a material's structure. Reproduced from Cranford et al. 2012.

a materiome and how they combine together to create the remarkable structures that define biomaterials.

One example of such a structure is cancellous bone, the type which makes up trabeculae, in which rods and plates form a connected network of either open or closed cells (Gibson 1985). Figure 1.3 is an SEM micrograph of cancellous bone along with a trabecular structure generated via genetic algorithm for comparison. Since the 19th century, scientists have observed that the orientation of struts align with the orientation of the principal stresses (Turner 1992). These observations led Julius Wolff, in 1892, to hypothesize that bone is a structure with a self-optimizing capacity (Wolff 1892). Bone has the ability to dynamically

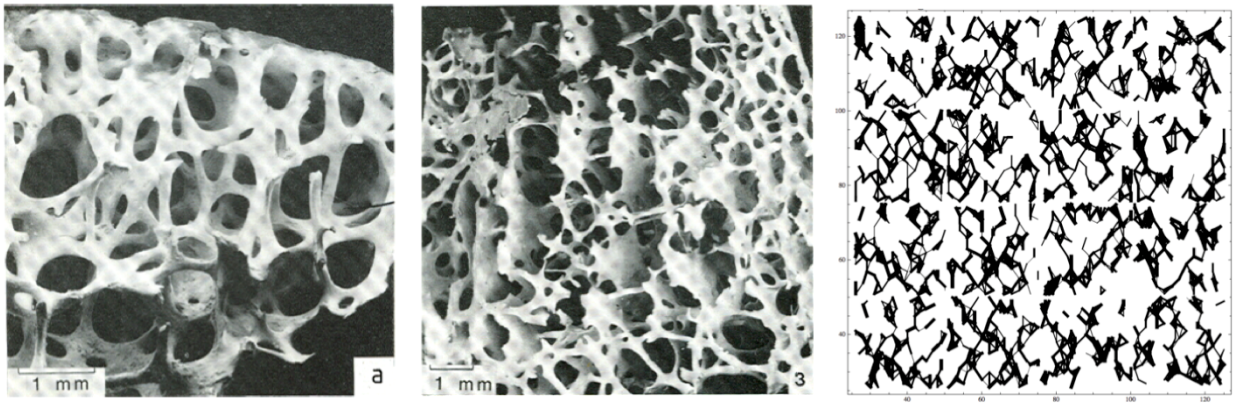

Figure 1.3: (a) SEM micrographs of cellcular cancellous bone structure reproduced from Gibson 1985. (b) Trabecular microstructure generated via genetic algorithm.

evolve its microstructure in order to adapt to changes in the loading conditions it experiences; changing the magnitude and orientation of the stresses a bone experiences will cause changes in the orientation and thickness of the struts within the cancellous region.

When bone initially forms, its struts are orientated is in the same direction as the direction of the average principal stress (Carter et al. 2001). The development of bone is strongly influenced by the local stress history, and when the stresses change the architecture and density of bone can change significantly in response (Carter et al. 2001). There is feedback between the structure of bone and its loading pattern; changes in loading initiate changes in structure, and developed adult bone has an internal structure tuned for its customary mechanical loading (Carter et al. 2001).

Several experimental studies have demonstrated the connection between the microstructure of cancellous bone and the loading patterns. Work on guinea fowl in 2006 showed that when the birds ran on an inclined treadmill, the orientation of their trabecular struts changed to match the angle of the incline (Pontzer et al. 2006). This suggests not only that the trabecular architecture is highly sensitive to its environment, but also that differences in architecture can result from differences in loading conditions.

Other studies have used differences in the trabecular architecture to account for different biomechanical adaptations. One study found that differences in trabecular architecture could account for why humans are susceptible to fractures of the vertebral body, while apes are not (Cotter et al. 2011). Other researchers have used trabecular architecture to demonstrate that early human ancestors were bipeds (Galik et al. 2004, Rook et al. 1999). The explanatory power of bone architecture extends beyond humans; researchers showed how the evolution of dogs for either fighting or running resulted in different bone architectures (Kemp et at. 2005).

Given the connection between bone architecture and stress history, it is clear that bone represents a highly responsive material, able to reorient and adapt its microstructure to any possible loading conditions. Through this dynamic evolution, bone is able to obtain maximum mechanical efficiency with minimal mass. The purpose of this thesis is to model the microstructure evolution of bone using the principals of cumulative selection—the same mechanism through which bone developed its ability to evolve its microstructure. Although this work focuses on cumulative selection as it applies to bone, it also aims to establish cumulative selection as a general principal for material's optimization.

The next chapter will survey previous computational modeling efforts of bone microstructure evolution. Following that background information is a chapter on cumulative selection and genetic algorithms. After the background is a chapter that contains the details about how the trabecular genetic algorithm was created. The next chapter contains the results of the simulations using the genetic algorithm to model the evolution of different trabecular microstructures under different loads. The discussion section further unpacks the results of the simulations, including providing a comparison between the techniques of previous bone modeling work with the genetic algorithms. Finally, the conclusion places the results of this thesis back in the larger context of materials science and computational modeling.

Chapter 2

Modeling Trabecular Microstructure Evolution

Previous computational studies on trabecular architecture have focused on three areas: attempting to verify Wolff's hypothesis that bone is a self-optimizing structure, quantifying the features of a trabecular architecture that has fully adapted to its loading conditions, and connecting the cellular level remodeling process to the overall 3D macroscopic mechanical adaptation. Of the three objectives, understanding how macroscopic features like stress and strain drive microscopic changes in bone architecture has been a central question in a number of previous studies.

At the scale of individual trabecular struts, microstructure evolution occurs via surface remodeling by cells called osteoclasts and osteoblasts (Adachi et al. 1997). Osteoclasts absorb surface bone and osteoblasts form new surface. Through the cycles of osteoclasts and osteoblasts, with different proportion of time in osteoclasts to osteoblasts, macroscopic changes in trabecular architecture can occur (Adachi et al. 1997). However, a central assumption of this model is that the osteoclasts and osteoblasts can detect feedback—like number of stress cycles of a given magnitude or local stress—of local from the overall trabecular architecture

and adjust their proportions accordingly.

Previous studies have explored several possible feedback mechanisms that can drive trabecular remodeling. Common driving forces include the ratio between local and average stress on a trabecular strut; comparing the local and global strain energy density; and using a minimum number of stress cycles of a given magnitude (Adachi et al. 1997, Huskie et al. 2000, Bagge et al. 200, and Carter et al. 2001). In all these studies, the driving force is related to a rate remodeling equation which determines whether to add or remove surface elements to individual trabecular struts. The equilibrium state is therefore the architecture where the driving force, and thus the remodeling rate, on all struts is zero. In non-equilibrium conditions, material is added or removed from individual struts until a steady state solution is reached (Adachi et al. 1997). One key feature of the remodeling rate model is that there is no global optimal solution; the model only describes the behavior of individual struts (Adachi et al. 1997).

Besides using a remodeling rate equation to capture the relationship between cellular remodeling and the macroscopic stress state, researchers have also used topology optimization to model the process of trabecular adaptation. Topology optimization calculates either the stress or energy of individual struts and adds material to struts with high energy or stress and removes material from low stress or energy struts (Jang et al. 2008). Although topology optimization and a rate remodeling equation appear to be distinct methods for describing trabecular architecture adaptation, it has been shown that both methods produce equivalent solutions (Jang et al. 2008).

Whether using a rate remodeling equation or topology optimization, previous work has all used Finite Element analysis. In the Finite Element studies, trabecular struts are modeled as a discrete number of pixels in either 2D or 3D (Adachi et al. 1997). The trabecular architecture is then an array of individual trabecular struts with fixed boundary conditions. During the simulations, a stress is applied somewhere in structure, and the local stresses

of all the surface pixels are calculated using Finite Element analysis. Then using either topology optimization or the rate remodeling equation, surface pixels are either added or removed according to the specific criterion of the study (Adachi et al. 1997). This is an iterative process that continues until either an optimal topology is found or the remodeling rate on every surface pixel is zero.

The results of the previous computational bone modeling studies have all supported Wolff's Hypothesis (Adachi et al. 1997). The results of the simulations were able to capture the functional adaption of trabecular structures—demonstrating the ability to connect cellular processes to macroscopic stimuli—and they showed good agreement with experimental observations (Tsubata et al. 2009 and Boyle et al. 2011). Additionally, the showed several important features of the equilibrium structures: trabecular struts aligned in the direction of the largest principal stress, and that the thickness of a strut increases as a strut aligns with the principal stress (Adachi et al. 1997).

Another important observation is that the equilibrium structures exhibit a uniform stress distribution and a uniform strain energy distribution. This implies that each strut is used equally to support the load, that the structure is fully stressed (Boyle 2011). This is one condition for maximum mechanical efficiency, and demonstrates that bone is in fact a self-optimizing structure.

The previous studies on the microstructure evolution of trabecular architecture have shown that external loads can drive cellular level remodeling that results in a trabecular architecture that has been optimized for that load, one in which each strut is used equally to support the load, indicating a fully stressed structure. In contrast with previous studies, modeling trabecular microstructure evolution using a genetic algorithm does not use topology optimization or a rate remodeling equation and creates a new structure with each generation, rather than adding or removing material to the existing structure. Instead, it optimizes through cumulative selection. To fully appreciate the distinction between this new model

and previous work, the nature of cumulative selection and genetic algorithms are described in the next chapter, after which the genetic algorithm for trabecular microstructure evolution is created.

Chapter 3

Genetic Algorithms and Cumulative Selection

3.1 The Power of Cumulative Selection

One of the most remarkable aspects of biology is the range of conditions in which living things have adapted to survive, in deserts, the Arctic, and even near boiling vents at the bottom of the ocean. In each case, organisms have adapted by developing specialized structures optimized for the functions that the organism needs to survive in the environment in which it finds itself. Perhaps the most amazing aspect of this process is that it relies on a single design principle: evolution through natural selection.

According to Richard Dawkins, evolution is the process of cumulative selection, directed by non-random chance of survival (Dawkins 1996). The power of cumulative selection is that it is not a random process, even though the mechanism for generation diversity, genetic mutations, is random. In evolution, cumulative selection is non-random because the only mutations that propagate across generations are those that increase an organism's probability of reproduction.

The other aspect of cumulative selection that makes it such a powerful method for adaptation and optimization is that each beneficial adaptation builds upon the previous ones. This is because the each new generation begins with all the traits of its parent generation. In other words, each improvement in an organism's ability to reproduce in its environment is used a starting point for future adaptations (Dawkins 1996).

Using cumulative selection, organisms have adapted to survive in almost every possible environment, demonstrating the breadth of the design principal of cumulative selection. The reason for this is that there is never a long-term selection criterion that pushes a species to adapt in particular direction. Instead, the selection process is always short term: better odds of survival or reproductive success (Dawkins 1996). This means that when an environment changes—and what structures or traits are necessary for reproductive success in that environment changes—the species in the environment can readily adapt to the new challenges they will face.

Biology has demonstrated the power of cumulative selection as a tool to evolve and adapt structures that allow organisms to thrive in their environment. Materials Scientists also try to design structures that allow for specific functions and that meet goals in performance. Just like Mother Nature, Materials Scientists can use cumulative selection to evolve initially random structures into structures that are optimized for a specific function. Unlike the process of natural selection, however, Materials Scientists can perform cumulative selection with a long-term goal. Using this design paradigm and computational modeling, Materials Scientists should be able to create structures as specialized as Mother Nature, but without having to use millions of years of evolution.

3.2 How Cumulative Selection Works

Cumulative selection, whether in biology or in materials science, is a simple process using three steps: development, selection, and reproduction. Through the repetition of theses three steps across multiple generations, a random starting point can lead to a structure highly optimized for an application or environment. One important key though is that a structure is not what is passed on from generation to generation. What gets reproduced and passed down from parent to offspring is instructions for how to make that structure, a set genes, or an entire genome.

The interesting thing about this mechanism is that although genes are what is passed down through generations, genes are not what is selected for during cumulative selection. It is the effects that genes produce on organisms—the proteins, structures, and behavior for which they encode—and whether or not these things increase the odds of survival and reproduction that determine whether or not a gene will be selected for and propagate across generations (Dawkins 1996).

The first step in cumulative selection is the translation of the genome into the structures that make up a living entity. This is the process of development, where the instructions for creating proteins, cells, and life are decoded from the genome an organism inherits from its parents. The initial development process is completely deterministic; the final structure of an organism is completely determined by its genes, and identical genomes produce identical structures. Over time though, environmental input may cause changes in the structure independent of the genes.

The next step in the process is selection. In natural selection, the criterion is whether an organism lives long enough to reproduce. When the cumulative selection process is used in other applications, like materials science, there may be some objective function that is used to quantitatively compare a set of developed structures. In that case, only the genes of

the optimal structure will be selected for and will enter into the reproduction phase of the process.

In reproduction, the genome of the selected organism is taken as an input and copies are made. The key feature of reproduction though is the possibility of mutation. It is mutations in the genome, which result in changes in the developed organism or structure, that allow for the adaption that defines cumulative selection. A mutation consists of a random change in one or more of the genes in the genome of an organism.

Evolution consists of the repetition of development, selection, and reproduction, with the endpoint of one generation as the starting point of the next. The difference between the genome, and consequently the developed organism, of one generation and its offspring is small, but across many generations the differences aggregate. Furthermore, even though the changes in a genome between one generation and the next are due to random mutations, the long-term changes in the genome are not random. The generational changes in a genome are due to the selection of genes that develop properties that are useful for survival (Dawkins 1996).

3.3 Cumulative Selection and Genetic Algorithms

Cumulative Selection is an optimal design method when trying to create structures optimized for a specific function without any a prior knowledge about what form the optimal structure will take. Cumulative selection makes no assumptions about what the optimal solution looks like; instead it allows for gradual improvements to be made to an initial random structure until the performance of said structure meets some predefined criteria. For Materials Scientists seeking to create new types of structures with novel functions, properties, and state of the art performance, cumulative selection offers a simple method capable of designing structures for numerous applications. A genetic algorithm is a computer program that uses

the process of cumulative selection to design such structures.

Just like cumulative selection used in biology, a genetic algorithm optimizes a genome based on a program to translate the genome into a developed system and then select for an optimal system based on some criterion. A development function translates the genome into its developed counterpart, an organism with proteins and cells in the case of biology. The genes that make up the genome are the degrees of the freedom that the algorithm can change.

Once a genome and a development function are defined, all possible structures are defined as well, at least in a mathematical sense (Dawkins 1996). In practice however, for large genomes, it is essentially impossible to comb through all possibilities to find an improved outcome. Because of the impracticality of testing every possible genome, it is almost always impossible to know whether a given outcome is in fact optimal. Instead of searching for an optimal configuration, the goal of the algorithm is generally to evolve until some increase in performance as been reached. The power of a genetic algorithm comes from its ability to efficiently search through different outcomes until it finds one that reaches a performance goal. (Dawkins 1996).

Another useful aspect of a genetic algorithm is its ability to allow for the characterization of precise relationships between developed structures; every genome occupies a unique point in genetic space, defined as the phase space of all possible genomes, and there is a unique spatial relationship between the every genome in genetic space. The relationship between two developed structures can then be characterized by the euclidean distance between the genomes for those structures in genetic space. This allows for precise quantification of the evolution of a genome over the course of a genetic algorithm program.

Genetic algorithms can be developed for any application. As long as the features of a system can be encoded in a genome in such a way that each genome represents a unique developed structure, a genetic algorithm can be created. Then, once a selection criterion is

defined, the genetic algorithm program can be run and will evolve an optimized genome from a random starting point. The result is the efficient evolution of a system from a random starting point towards optimized conditions.

3.4 A Genetic Algorithm for Radiating Lines

In order to gain experience with genetic algorithms, I decided to create a genetic algorithm that would develop and select for images of lines radiating from different points in 2D space. This is a useful exercise because even though radiating lines are unrelated to trabecular bone, the code and sections of the algorithm will be analogous. By first developing an algorithm for a simpler system, I will learn how best to structure the code and also about how to think about the processes of development, selection, and reproduction. In addition to helping understand how to create a genetic algorithm, the radiating line algorithm will also validate the general nature of this method of optimization. In short, this simple case study offers a chance to better understand the key features of a genetic algorithm and how it is able to optimize properties by mutating genes and changing structure.

In my example, I considered a genome that, when developed, would produce different numbers of different length radiating lines from different points in space. Figure 3.1shows a translation of one such genome. The number of lines, the length of the lines, and the points in space from which the lines are drawn are all determined by the genome for this code. Those three input genes characterize this simple algorithm. Table 3.1 shows a subset of the genes for this algorithm along with an example genome and it's transition.

Each of the three genes is one number from a different discrete set of possible values. The number that defines a gene corresponds to a position in a list of either a number possible lengths, a list of the number of lines that are generated, or a list of the points in space from which lines are drawn, In a compromise between the complexity of the system and the

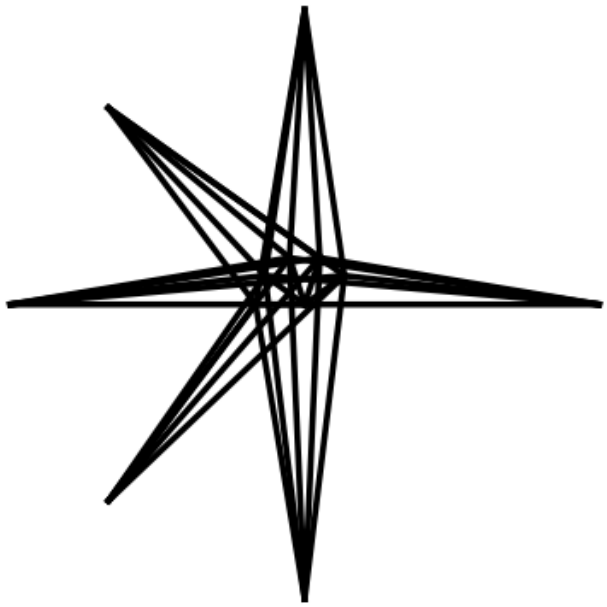

Figure 3.1: An example image of the types of radiating line pattern the genetic algorithm can produce.

Num. of Points	**Num. of Lines**	**Line Length**
2	1	0.25
3	2	0.5
4	3	1
5	4	1.5
6	5	2

Example Genome: (3 2 4) *Translation:* 4 points, 2 lines, 1.5 line length

Table 3.1: Partial range of genes for radiating line genetic algorithm.

computational burden, I choose nine possibilities for each of the three genes. This means that there are 729 possible genomes for this system.

After defining the genome for this algorithm, the next step was to create a development function, which takes a genome as its input and returns the translation of that genome into an image of different number of lines of different distances emerging from different points in space. For example, the image in figure 3.1 is the translation of the genome (4,4,5) where the first element is the length gene, the second is the number of lines gene, and the third is the points in space gene.

The next step was to create a reproduction function. A reproduction function takes in a genome and a mutation rate as its input and returns either the same genome or that genome with one or more mutation. My methodology for determining whether or not a mutation should occur was to generate a random real number between 0 and 1. If the random number is greater than the mutation rate, a parameter in the algorithm, then that gene was replaced by a random integer between 1 and 9, representing a different element in the list of possible genotypes for that gene. Assuming that the random real number generator generates all possible numbers with equal frequency, this method ensures that the probability of a mutation increases in proportion to the mutation rate. This method represents only one possible method for determining when a genome is mutated and other genetic algorithms may use a different mechanism.

The last step in my genetic algorithm was to create a selection criterion and a selector function. The selector function takes in a list of genomes as its input, uses the development function to translate those genomes into the series of lines they represent, evaluates the translations based on the translation criterion, and returns the genome that results in the translation that scores the highest according to the selection criterion. For my algorithm, I chose to select for genomes that produce the series of lines with the largest standard deviation in their length. This criterion was completely arbitrary, but I thought it would generate interesting results.

Once I had defined a quantitative selection criterion, all the possible genomes could then be ranked according to their average standard deviations. Because there were only 729 possible variations, this was not a computationally intensive calculation. I was able to sort through all my genomes and find the optimal one, shown in figure 3.2. The optimal result has the genome (9,1,8) and results in an average standard deviation in line length of 2.20. Although I was able to easily determine this, in more complex algorithms with more degrees of freedom, it is inefficient to determine this genome by brute computation.

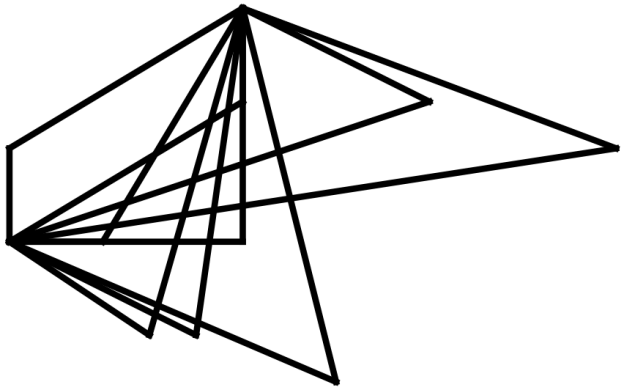

Figure 3.2: The optimal radiating line structure based on the selection criteria of the maximum standard deviation in line length.

Furthermore there is no guarantee that there is only one optimal is optimal; for more complex systems there may be multiple genomes which generate optimal properties. A better strategy therefore, is to use the algorithm to preform genetic evolution.

The last step to using the genetic algorithm was to combine the developer, selector, and replicator into a single evolution function. The evolution function takes as its input an initial genome, a mutation rate, a number of offspring produced during reproduction, and the number of generations of evolution. The evolution function then returns a list of genomes, each one corresponding to the genome that was selected in that generation.

Once I had the evolution function, I could use it to evolve random initial genomes into optimized genomes. Because I knew both the optimal genome and standard deviation it produced, it was easy for me to determine whether or not the algorithm was working correctly.

What I did in order to gain insight about genetic algorithms was to run my algorithm with the same initial genome and three different mutation rates, a high rate of 50 percent, a low rate of 25 percent, and a medium rate of 10 percent. Because the algorithm has a random nature to it, I needed to look at averages across runs in order to get a better assessment of the performance of the code. I decided to run the algorithm 10 times for 50 generations at each mutation rate. The results for one of those ten runs are shown in figure 3.3.

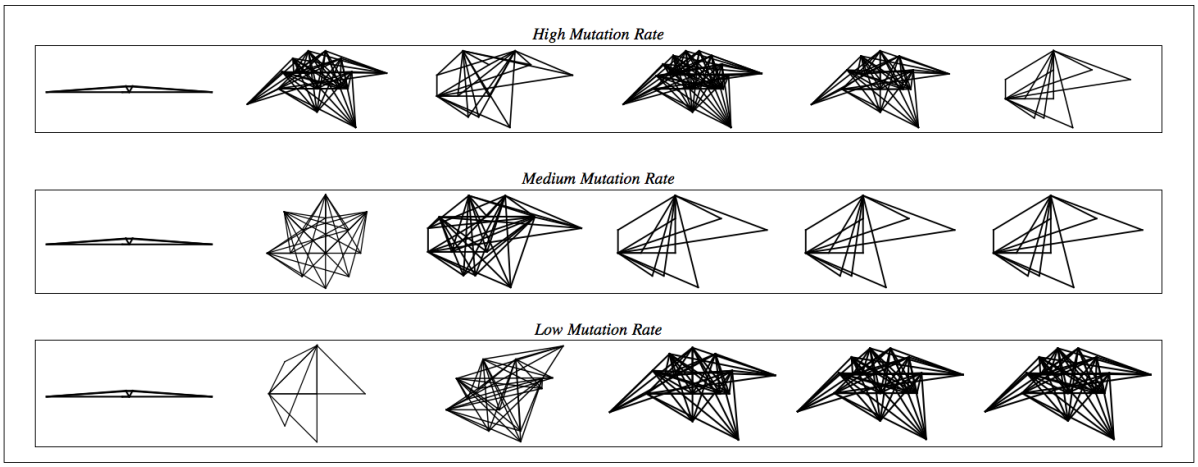

Figure 3.3: The evolution of one set of outputs of radiating line structures for a high, medium, and low mutation rate. Shown are the first, 10th, 20th, 30th, 40th, and 50th structures that were generated by the algorithm. In the medium and high mutation rate cases, the optimal structure was found by the end of the 50 generations.

This method also allowed me to gain insight about the effect of mutation rate on the performance of the code. I took the average of the standard deviation in line length, the selection criteria, across all 10 runs for each of the fifty generations in all three cases, low, medium, and high mutation rates. A plot of those results is shown in figure 3.4.

The plot shows several important trends. First, in all three cases, the average standard deviation increased, showing that the algorithm was working correctly. The second trend is that the rate at which the average standard deviation increased depended on the mutation rate, with a higher mutation rate resulting in a faster increase. The downside of this is that a higher mutation rate also could lead the system to regress once it was close to the optimal conditions. This trend was only present in the high mutation rate case. In the case of a medium and low mutation rate, improvements were slower but stable. This implies that there is an optimal mutation rate where the system quickly evolves towards the optimal configuration and all improvements are stable. In the case of the trabecular genetic algorithm,

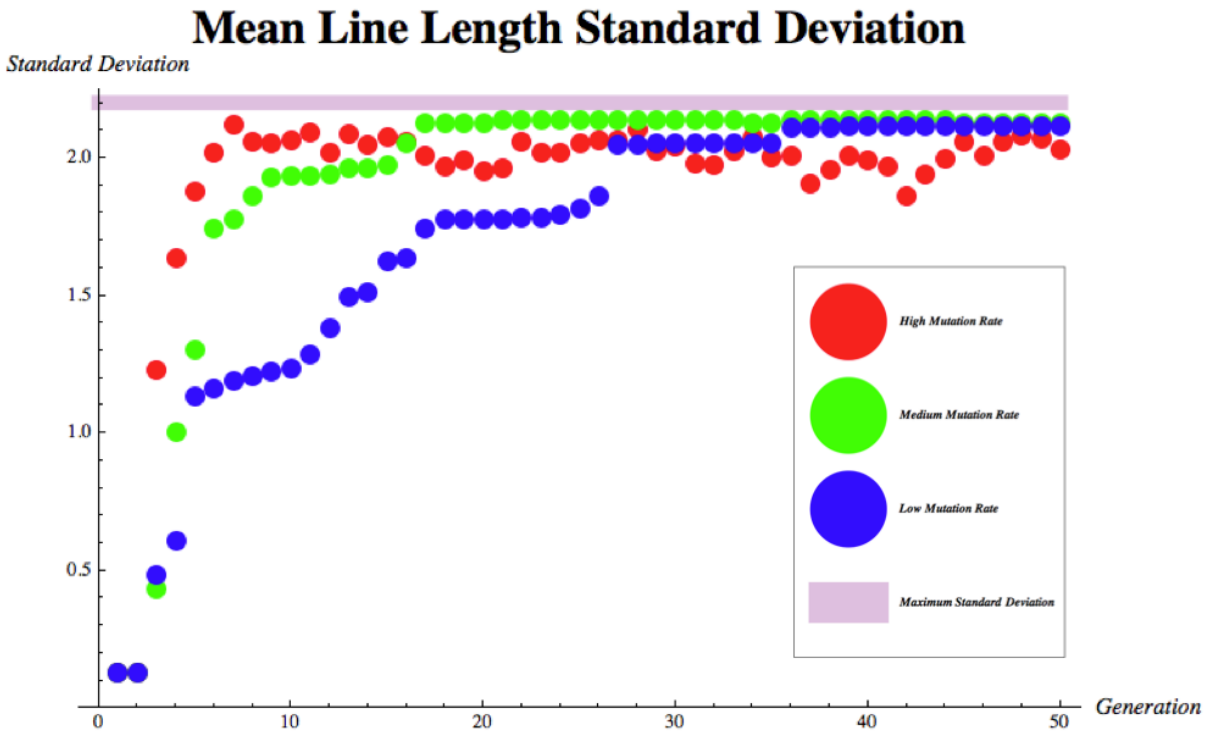

Figure 3.4: A plot of the average standard deviation vs. generation number for 10 runs each at a low, medium, and high mutation rate. The mean standard deviation increases at a faster rate with a higher mutation rate. However, at high frequencies mutations are more likely to cause a decrease in standard deviation. Lower mutation rates are less likely to lead to regression.

simulations will be conducted at multiple mutations rates to determine the sensitivity of the algorithm and the optimal rate.

The radiating line genetic algorithm was an instructive way to learn the general principals behind genetic algorithms. I will apply the same principals when I create a genetic algorithm for trabecular structures. The code will have a very similar structure. The main differences will be in the complexity of the trabecular genome and in the selection criteria. The lesson I learned about genetic algorithms—the effect of mutation rate on performance—will also apply in the case of trabecular structures. For those two reasons, this served as an excellent introduction into genetic algorithms and has provided valuable insights that will guide me in the more complex problem.

Chapter 4

Creating a Genetic Algorithm for Trabecular Microstructure Evolution

In order to design a genetic algorithm to simulate the microstructure evolution of trabecular bone for a given stress state, it is first necessary to consider the features that define a trabecular architecture, and the degrees of freedom necessary to accurately model the microstructure evolution and optimization process. The genetic algorithm will use the same principles described in the previous chapter. This time, however, instead of having genes that translate to a radiating line pattern, the genes will translated into the defining features of a trabecular structure.

The genes for the trabecular genetic algorithm will correspond to the degrees of freedom in the overall architecture. Once these genes have been defined, the development function will translate a trabecular genome into a microstructure. The microstructure will then be placed under an applied load and its response will be calculated. Using the response to the applied load, a selection function will choose the genome that translates into the best trabecular architecture for those conditions. That genome will then go through reproduction where it will be copied and mutated. The offspring of the genome will then be placed under the same

load, and the offspring that best matches the selection criterion will be selected. Repeating this process for multiple generations defines the evolution of a trabecular architecture and the trabecular genetic algorithm. The first step in the process is to define the features of trabecular microstructure and translate them into a genome.

4.1 Characterizing Trabecular Microstructure

In 2D, a trabecular microstructure consists of a network of struts of different length, angle, and orientation. Those three features represent the degrees of freedom for the individual struts that together define the microstructure. At the microstructure level, the defining feature is the density, which can be defined several ways. For this genetic algorithm, the number density, the number of struts per unit area, will be the defining criteria.

As previous studies have shown, during the microstructure evolution and optimization of a trabecular architecture, the thickness, length, and orientation of the struts change in response to the loading conditions. In the algorithm, these changes are modeled by changing the genome that determines the type of struts that are constructed in a region.

An important feature of a trabecular architecture is the emergence of different domains, characterized by a local strut angle, thickness, or orientation. In order to capture that feature in our algorithm, the overall trabecular structure will be divided up into sub-sections, each of which will have its own independent genome. The overall architecture is then given by the combination of all the different sub-sections into a larger system. For this algorithm, the sub-sections were chosen to be squares, but they can have any geometry. This allows for individual domains to adapt to local applied stresses without affecting other sections of the overall structure that may have a different local stress state. Furthermore, by dividing up the system into domains, it also allows the strut number density to vary locally, adding another degree of freedom to the algorithm, allowing more accurate modeling of the microstructure

evolution.

Given both the essential features of microstructure evolution and the degrees of freedom that define both the individual struts and a domain in the overall architecture, we can then define the genes and the genome that will form the base of our algorithm. The genome can be considered to contain information about multiple levels of the architecture. At the lowest level, it contains instructions about the types of struts that will be created in a region, the angle, thickness, and orientation. At the next level, it contains instructions about the strut number density for the local region. The overall genome for a trabecular architecture can be thought of as the aggregate of all the genomes for all subdomains within the boundaries of the architecture. A key feature though is that the genomes of the subdomains are independent; there is no global goal or driving force to create a unified genome across the system. Instead, the genome of a domain is determined solely by optimized the local microstructure for the local stress state.

4.2 Trabecular Architecture and Graph Theory

Given the model of trabecular architecture as a network of connected struts of different thicknesses, angles, and orientations, a trabecular structure can be modeled using the principles of graph theory as a network of vertices and edges. Within this framework, the edges represent the struts and the vertices are the points in space where two or more edges intersect. The distinction between this model of trabecular microstructure and graph theory is the geometry; the absolute locations of the vertices in space are a defining feature of a trabecular architecture, while the geometry of a graph can be changed arbitrarily without changing the features of that graph. Figure 4.1 shows an example two isomorphic graphs, graphs with different geometry but the same set of vertices and edges.

One of the most basic graphs is called a complete graph, with an edge between every

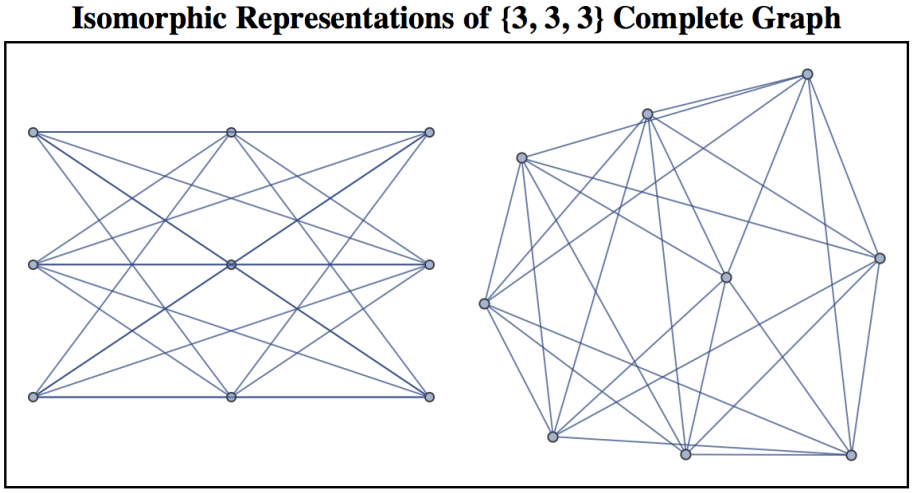

Figure 4.1: Even though the vertices of these two graphs are located at different points in space, the graphs are isomorphic because the edges between the vertices are the same for both graphs. Unlike these graphs, the location of the vertices is a defining feature of the trabecular architecture; changing the location changes the overall structure.

pair of vertices. Figure 4.2 shows a six vertex complete graph. All graphs, including the subset of complete graphs, can be characterized using an Adjacency Matrix. If there is an edge between vertices ij in the graph, then the adjacency matrix will have the value 1 at position ij and at ji, otherwise it will be a 0. In general, the adjacency matrix for a graph of N vertices is an N by N symmetric sparse matrix where the only non-zero values are at positions representing pairs of vertices connected by edges. An adjacency matrix can be un-weighted, where the non-zero entries are all equal to 1, or weighted, where the non-zero entries can take any range of values representing the weight of the edge between those two vertices.

Using the principles of graph theory to construct trabecular microstructure begins by creating a complete graph of N vertices with random edge weights. The edge weights will map to the thickness of the struts. The vertices will map to a set of points from which the struts will grow. The points onto which the graph will be mapped can take any form, such as a regular lattice or a random sample of points. Figure 4.3 shows a complete 100 vertex graph

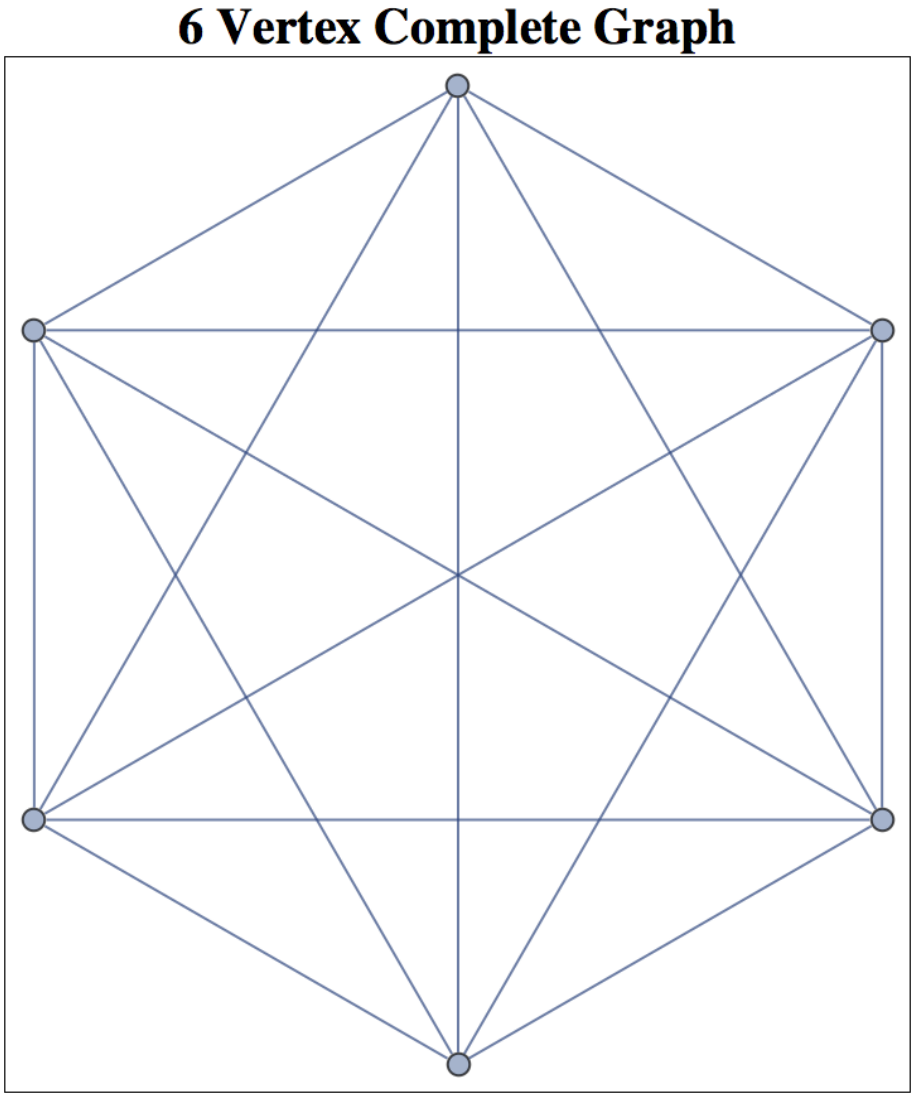

Figure 4.2: A complete graph is one of the most basic graphs, where every vertex is connected to every other vertex. A complete graph mapped onto a trabecular architecture would mean that every vertex connects to every other one, resulting in a number of struts equal to the sum from 1 to the number of veritices minus one.

with equal edge weights mapped onto a regular 10 by 10 lattice. Once a graph has been mapped to a lattice, the edges now take a definite geometry and can now be characterized according to their angle, length, and thickness, all which will eventually be determined by the genome. Before considering the strut characterization however, it is necessary to consider how to generate the set of points onto which the complete graphs will be mapped.

4.3 Generating and Developing Trabecular Microstructures

Assuming that the vertices from which struts originate can only be arranged in a regular lattice is unrealistic. A better approach would be to use a weighted random sample of initial points whose features can be controlled by another gene in the strut genome. To create such a random sample, random gray images were created where the pixel values were random real numbers from zero to one. Figure 4.4 shows an example of one of these images. Using the pixel values as a weighting function, a function then selects N pixels from the list with pixel values closer to one having a higher probability of selection. This method allowed for a more realistic point distribution than using a regularly spaced lattice. Figure 4.5 shows the results of this random point selection method for four different sets of points within a region of space. Each of the four regions selected a different number of total points from the random gray image. For trabecular structures, the number of points selected will be controlled using a gene.

Once a method for generating the set of points onto which the graphs were mapped was established, the next step was to reduce the number of struts from the complete set that the initial complete graph generated to a smaller subset that would be controlled by genes. The genes would control the number of struts that are chosen from the set of all possible struts and also the criteria for what struts are selected.

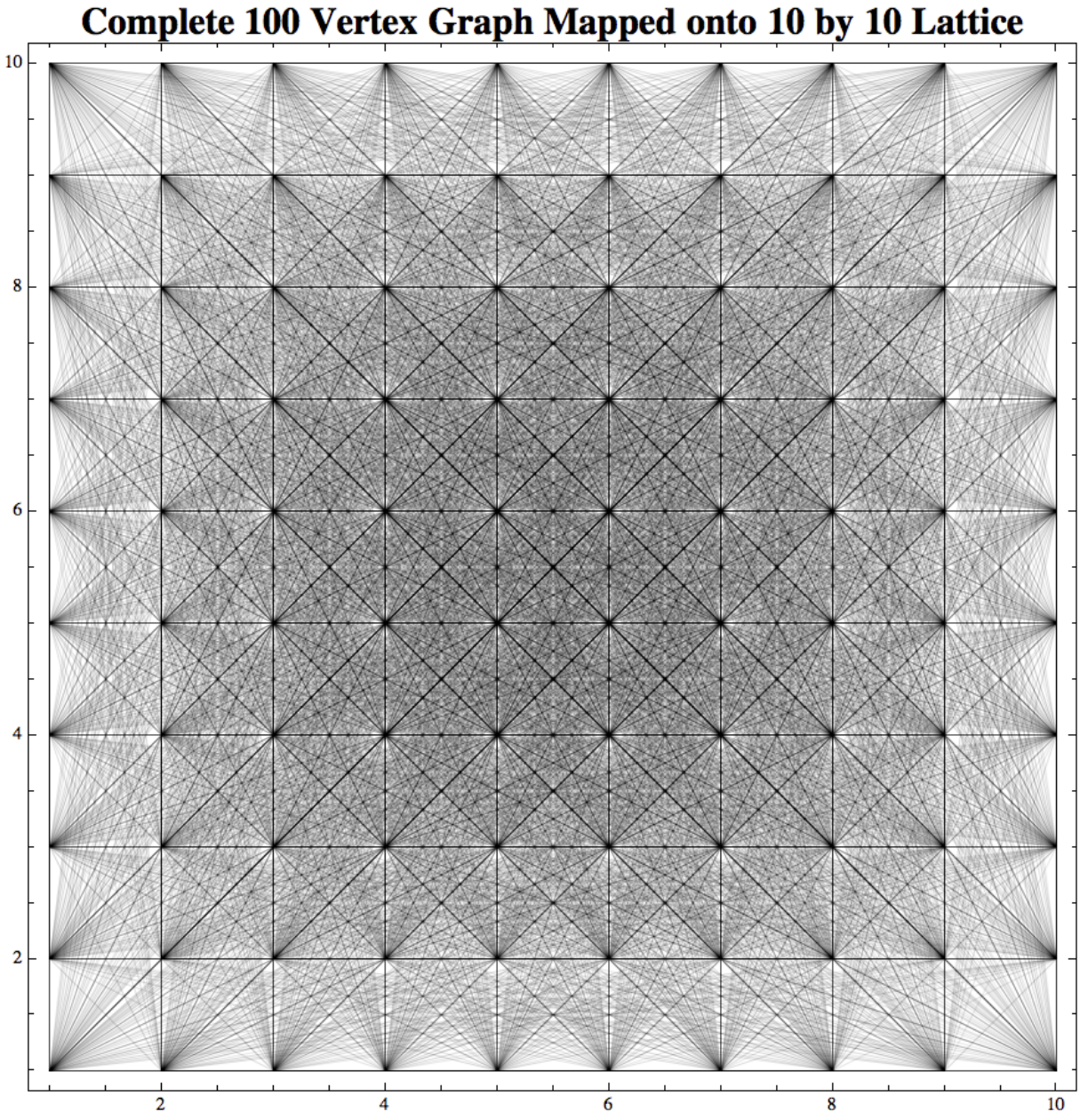

Figure 4.3: A complete graph with 100 vertices where each vertex is a point on this 10 by 10 lattice. Once the vertices and edges have been mapped to a lattice, they now be defined in terms of their geometry.

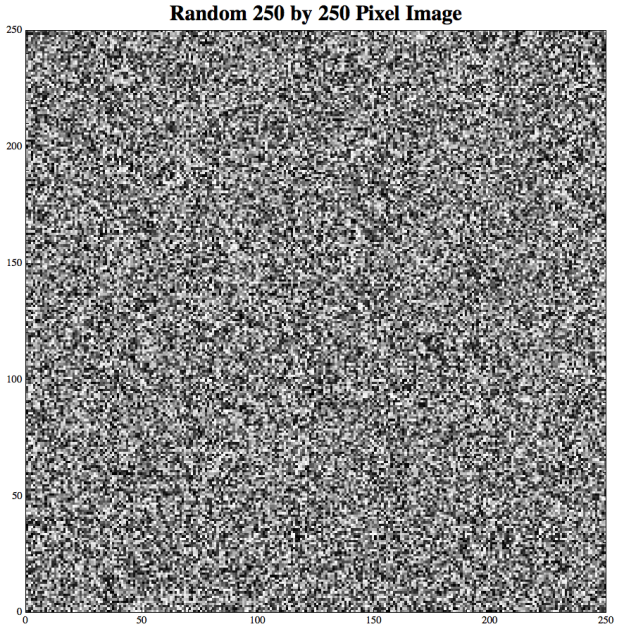

Figure 4.4: A random image that is used to generate the set of vertices within a trabecular cell. The pixel values in this 250 by 250 grayscale image, random numbers between 0 and 1, are used to determine a probability distribution when randomly selecting pixels to serve as vertices for struts. A pixel with an intensity closer to 1 is more likely to be selected, resulting a random weighted pixel distribution function.

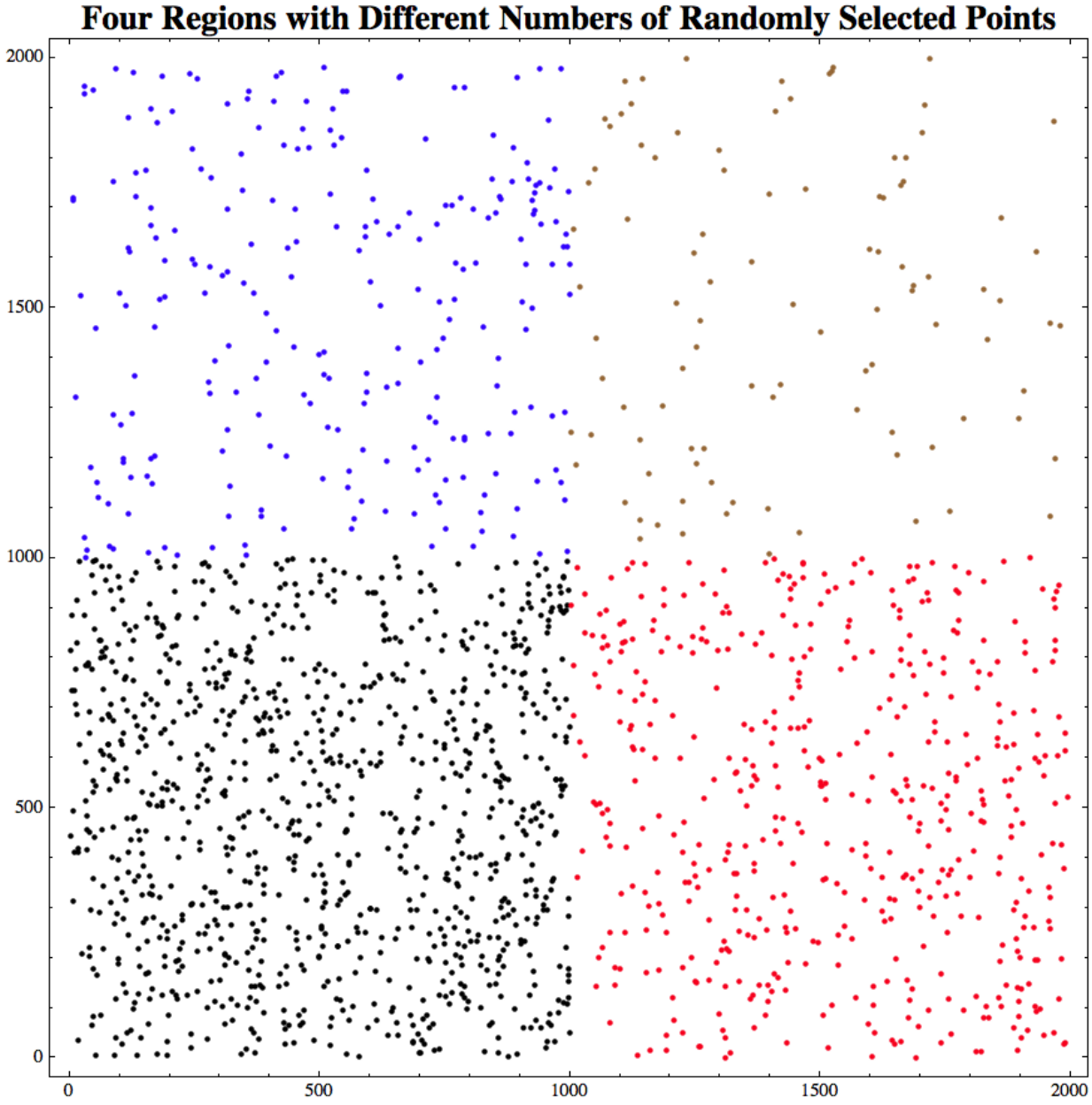

Figure 4.5: Each of the four colors represents the randomly selected points for a region of space from a random image like the one in figure 4.4. Each of the four regions has a different number of selected pixels. A trabecular architecture will have a gene for each region that determines the number of points that are selected from the random image. These points serve as the vertices from which struts can be created.

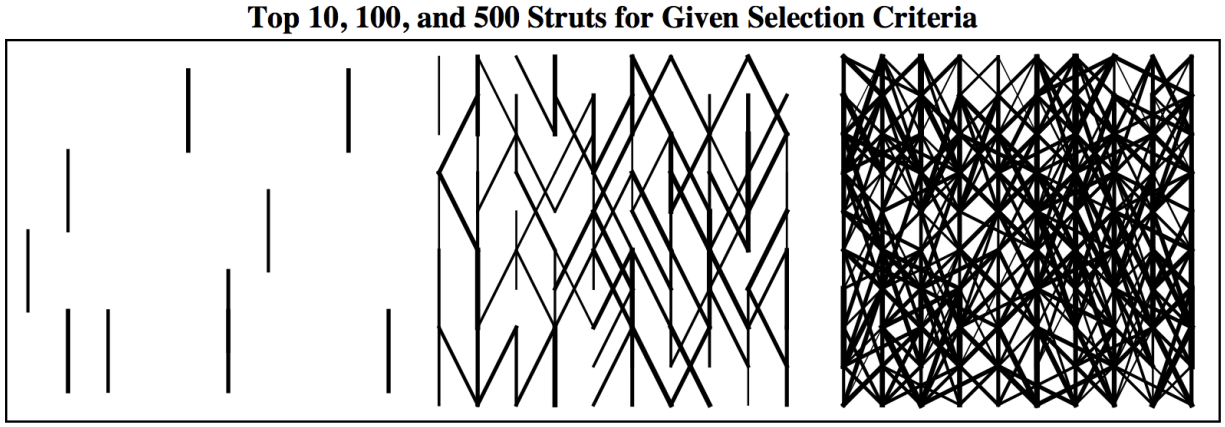

Figure 4.6: The top 10, 100, and 500, struts based on the results of the strut selector function. The struts were compared to a strut of length 1, with angle Pi/2, and of thickness 0.01.

The criterion for selecting a strut is a function of the geometry of a strut. Once the graph connections have been mapped onto a set of points, the struts take on a definite geometry, characterize by their length, angle, and thickness. By comparing these features of all possible struts to the features of a selection-strut, the struts can be sorted and ranked according to how close they are to the features of the selection-strut. The features of the selection-strut will be controlled by genes, which will determine its length, angle, and orientation. Another gene will determine the how the features are weighted; the simplest case is that all are equal, but a more complex gene could make the difference in angle between the strut and the selection strut twice, three times, or half as important as the differences in the other features.

Once the strut ranking function has been defined and a selection strut and weight function have been input, the strut ranking function outputs the list of struts sorted by how close they are to the weighed selection-strut. Next, a gene determines how many struts are selected for the final structure and the top N struts are chosen from the overall list are output. Figure 4.6 shows the results of the strut ranking and selector function for a simple regular lattice.

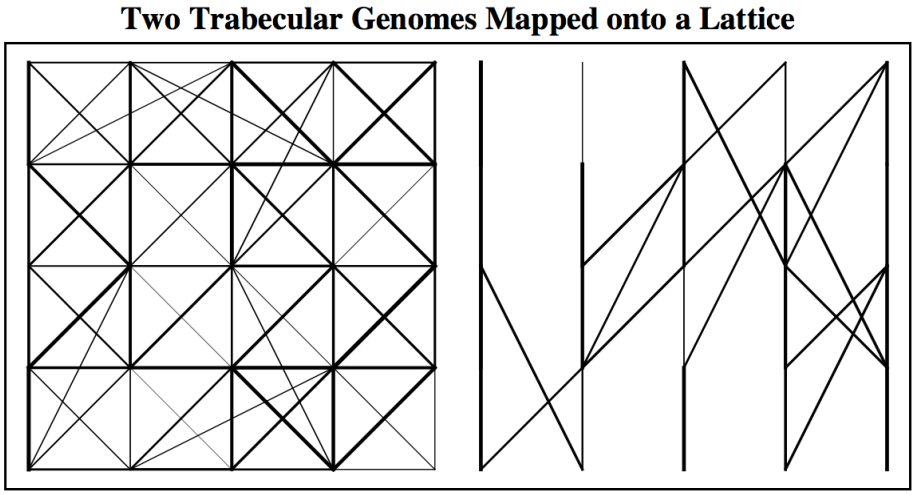

Figure 4.7: The results of translating a trabecular genome onto a set of lattice points. The genome specifies the number of struts that are created, the prefered angle, thickness, and length of a strut, and the weights given to each of those three strut features.

4.4 Trabecular Architecture Genome

After having created the functions that convert a complete graph into a 2D trabecular architecture, the next step was to create genes that can control the features of the architecture. The trabecular genome is a six dimensional genome that encodes information about all hierarchies in the trabecular architecture. At the highest level is a gene that controls the number of points that are selected as vertices in system. The next level is a gene that controls the number of struts in the system. At the bottom level are the genes controlling the features of the selection strut and the weight function. The weight function gene controls the importance of the differences between the struts and selection strut. The selection strut gene control the angle, length, and thickness of the selection strut, each of which can vary independently. Figure 4.7 shows the translation of two different trabecular genomes mapped onto a regular lattice.

An important thing to consider is that the genomes described above only characterize a subsection, or cell, in the overall trabecular architecture. Dividing the architecture into

domains is necessary to properly model ability of a trabecular microstructure to develop domains in response to local loading conditions. The genome for a whole trabecular system can be considered to be the list of genomes for each individual domain and the lattice points onto which each genome is mapped.

It is also important to make a distinction between the trabecular genome and the materiome. The genome is analogous to the DNA of a trabecular architecture while the materiome is genome along with the points onto which the genome is mapped and the instructions for translation. The distinction is in the perspective of the two view; in this case the genome is only the set of genes that determine the features of the strut, but the materiome is the entire process of trabecular development—the genes and the instructions for generating a trabecular architecture from those genes. In short, the materiome offers a complete picture of trabecular architecture while the genome only captures the features of the types of struts in each cell.

The first step to create a full trabecular architecture is to create the genomes for the individual domains. The second step is to take the random points gene and translate into a set of random lattice points. Once the lattice points for each region have been defined, the trabecular microstructure for the system has been determined. Before translating the random point genome, the microstructure has not yet been determined because due to the random nature of point selection the same point gene could produce a different set of points when translated different times, even though the macroscopic point distribution will be the same in both cases.

4.5 Generating Trabecular Architecture

Once the list of points and genomes is defined, they can be translated into the set of struts those genes produce. Figures 4.8-4.11 illustrate different structures produced under different

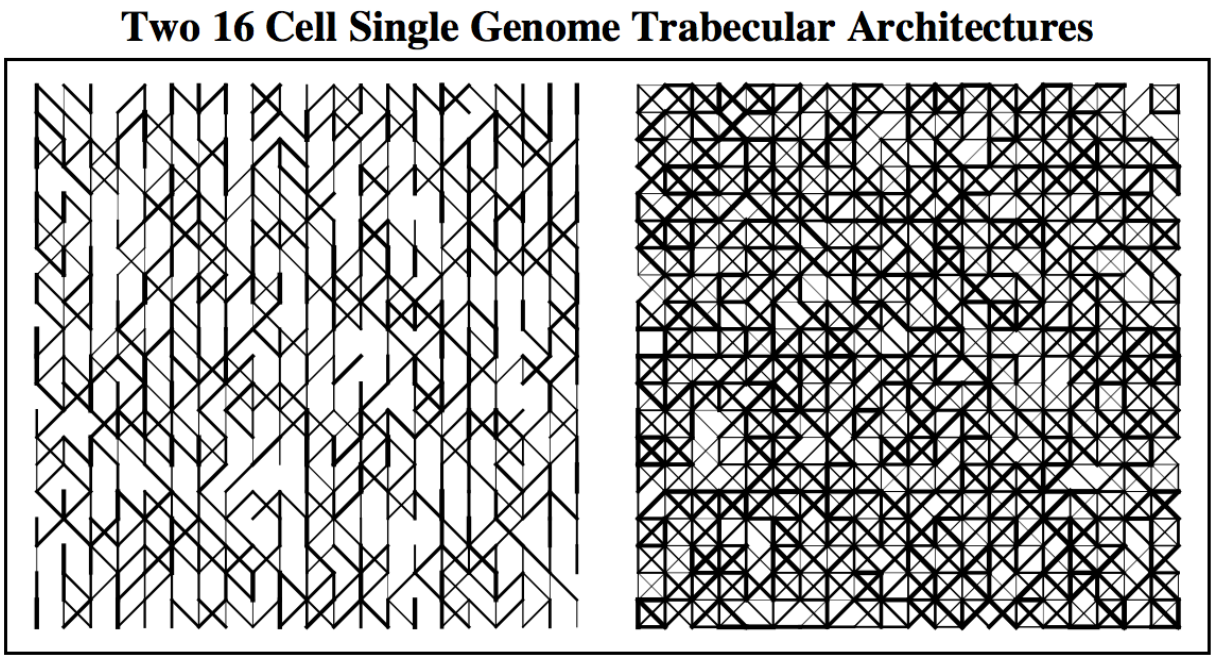

Figure 4.8: Trabecular architecture created by the combination of 16 different subcells, like the ones in figure 4.7. For these two architectures, the genome is the same for all the sub cells; however, the genome for the figure on the left is different than for the figure on the right.

conditions. Figure 4.8 shows two 16-cell trabecular microstructure mapped onto a regular lattice where each sub cell has the same genome. Figure 4.9 also shows a 16-cell trabecular microstructure where each cell has the same genome, but mapped onto randomly selected points instead of a regular lattice. Figure 4.10 shows another 16-cell architecture on a regular lattice, but one where each cell has its own unique genome. Finally, Figure 4.11 shows a 16-cell architecture for 16 unique genomes, but mapped onto a set of random lattice points. The gene that determined the number of random points for a cell was also different for each of the 16 regions.

The last step in the development stage of the genetic algorithm was to create functions that can determine the average features of a trabecular architecture. These functions will be useful when quantifying the evolution of a given genome under a specific load. Functions

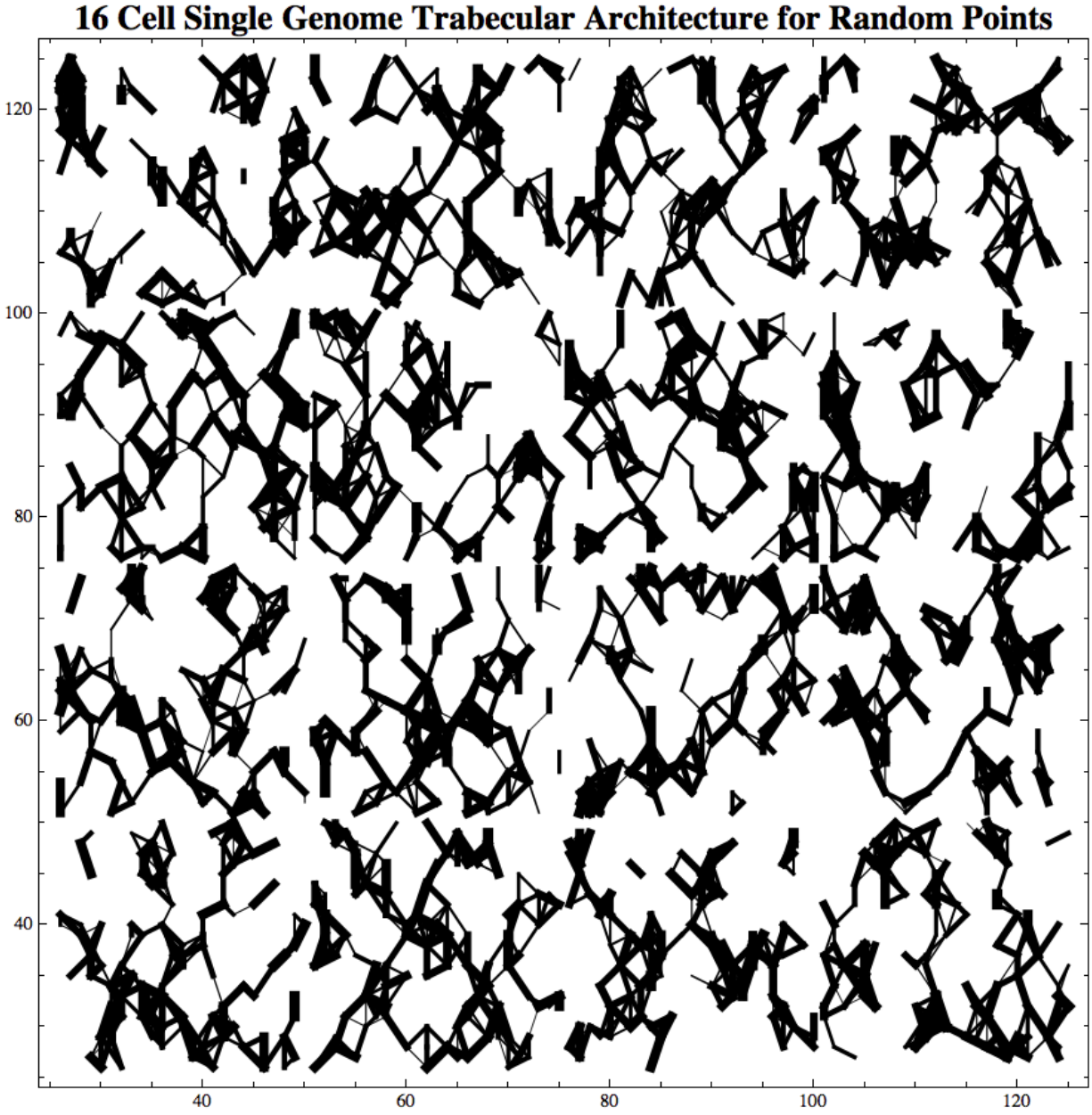

Figure 4.9: A translation for a trabecular genome mapped to a random set of lattice points. Each cell had the same gene that specified that 100 lattice points would be chosen at random. The genome also specified that 200 struts would be created in each cell, that the ideal strut had thickness 0.005, length 1, and angle of Pi/2. It also specified that the angle was 3 times as important as the length and thickness.

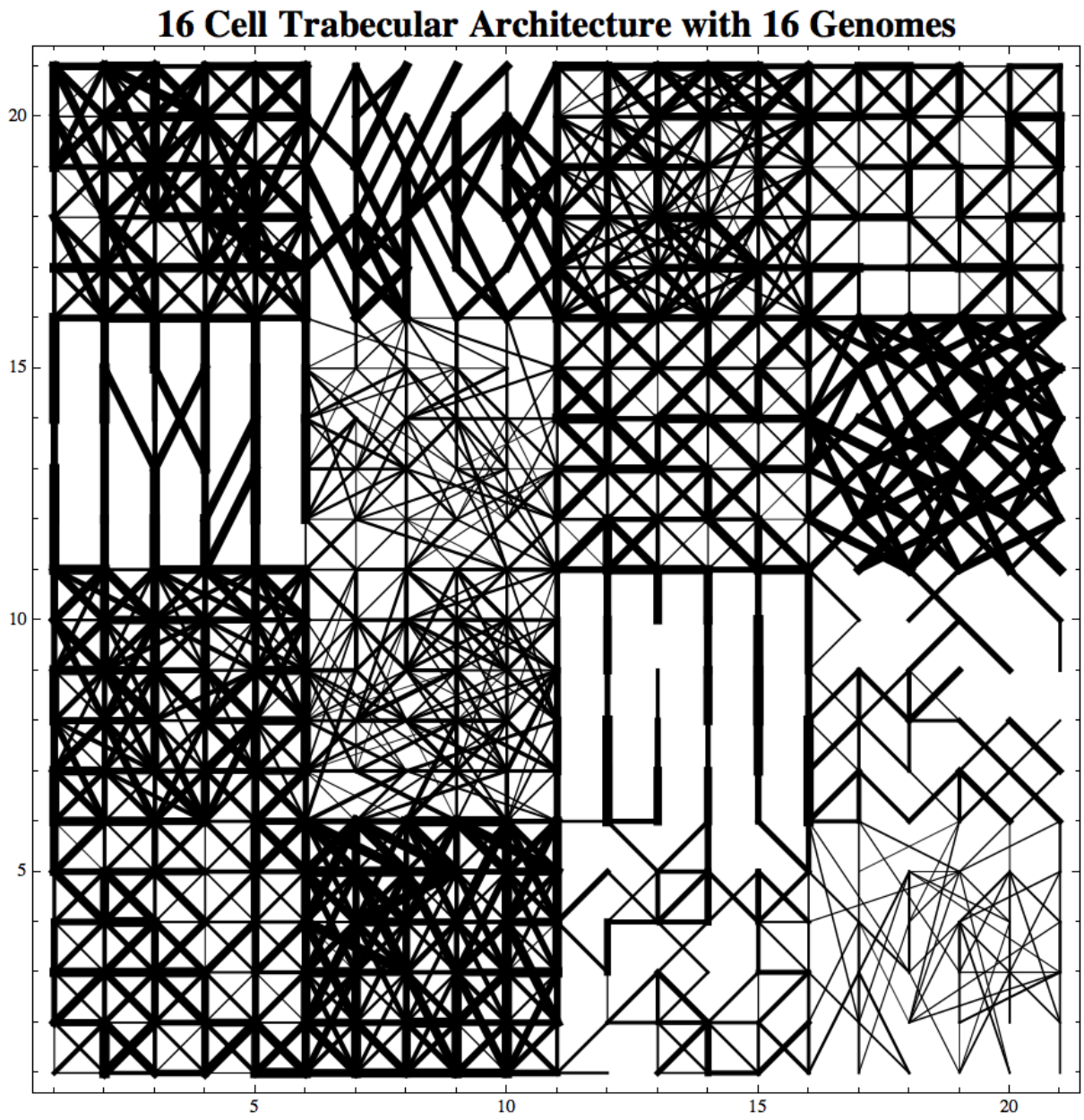

Figure 4.10: A visualization of a trabecular architecture made of 16 different genomes mapped onto 16 sets of lattice points.

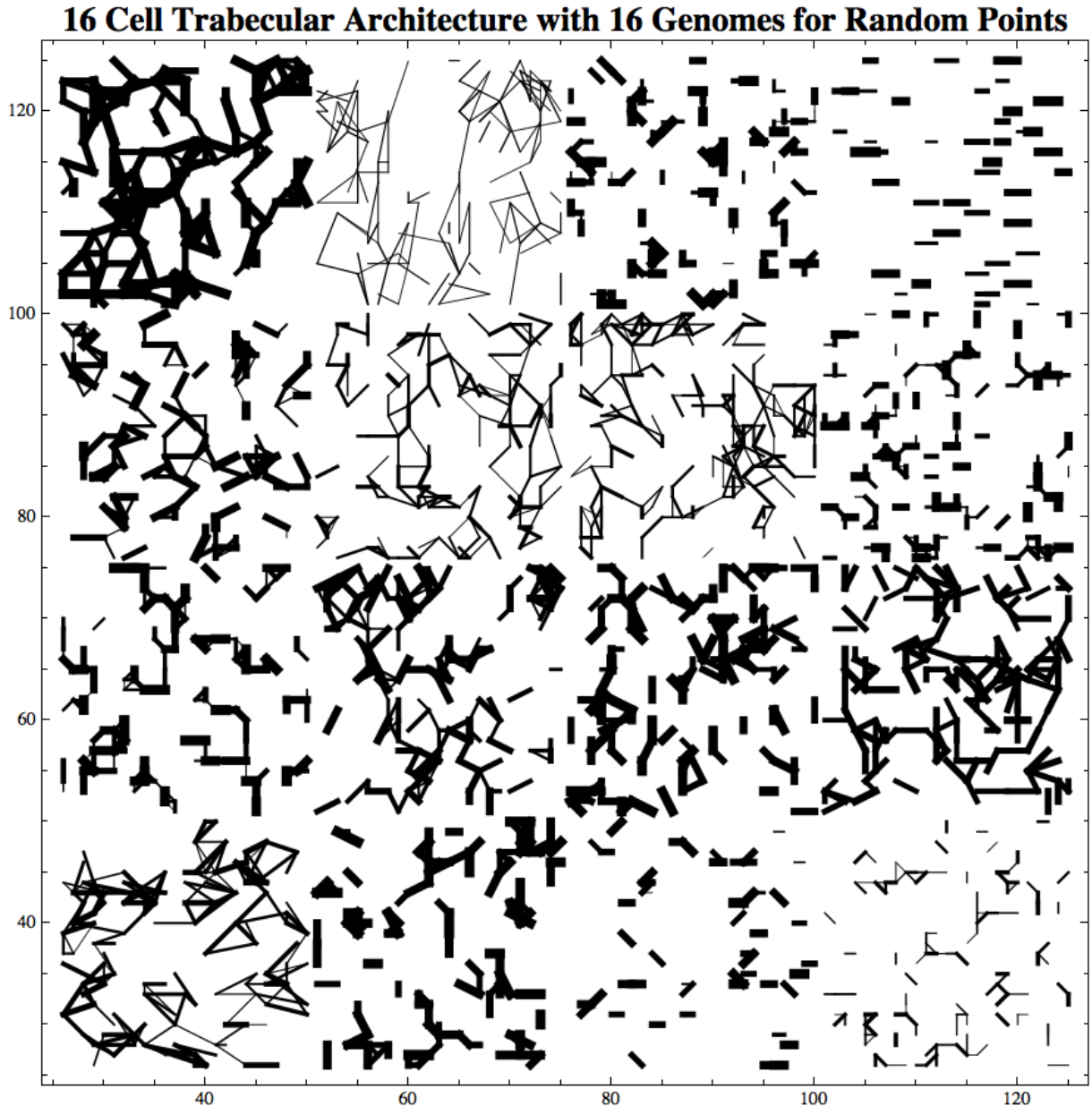

Figure 4.11: Trabecular microstructure for 16 independent genomes. Each cell has a unique genome that determines the number of random lattice points that were selected, the number of struts that were created, and the features of those struts.

were created that determine the mean length, angle, and thickness of the struts in a given cell and also in the system as a whole. The function also returns the standard deviation for those three features. These metrics allow for quantitative characterization of the microstructure evolution of the algorithm.

A final major advantage of creating trabecular architecture using the principles of graph theory is that these methods can easily be extended to 3D by mapping the initial complete graph to points in 3D space. Once the graph has been mapped onto the set of points, the same steps, with a few small modifications, can be used to produce 3D trabecular microstructure controlled by genes.

4.6 Trabecular Loading and Selection

As discussed in earlier chapters, an optimal trabecular architecture is one where the strain energy is equal for all struts. This implies that the stress on all struts is equal and that all the struts are therefore being used equally to support the load. By selecting for trabecular systems with the lowest standard deviation in the elastic strain energy, the system will evolve towards a set of struts that produce uniform stress for those loading conditions. In order to create this selection criterion, the first step is to create a function that calculates the strain energy as a function of the loading conditions.

Modeling the trabecular struts as beams, in 2D trabecular systems there are three types of stress under which the struts can be placed: a shear stress, a tensile stress, and a bending moment. The dimensionless strain energy of a beam under a given stress-state is given by equation 4.1:

$$U = \frac{1}{2}\frac{E(L_{cell}(\epsilon_{L,x} - \epsilon_{R,x}))^2 + \frac{E}{2(1+\nu)}(L_{cell}(\epsilon_{L,y} - \epsilon_{R,y}))^2}{EL_{strut}} + \frac{1}{2}\frac{EIw(\theta_L - \theta_R)^2}{E} \tag{4.1}$$

Where U is the dimensionless elastic strain energy, E is the Young's Modulus, L is the length of the cell or the strut, epsilon and theta are the strains of the left and right vertices of the strut, nu is the Poisson's ratio, and I is the moment of inertia.

When calculating the strain energy, it is important to consider how the connectivity of the system affects the strain energy. Whenever two struts share a common vertex, the displacement of that strut will change the energy of both struts. At each vertex there can be three different types of displacements, strains in the x, y, or theta direction, so the number of variables for the strain of a system scales as three times the number of struts.

The stress strut function uses equation 4.1 to calculate the energy in a strut as a function of its displacements. By summing the stress strut function over all the struts in a system, the total energy of a trabecular architecture under a given load can be computed. Minimizing this function results in the elastic strain energy of each strut in the system. Once the strain energy for every strut is known, the standard deviation can be calculated and the selection function can choose the system with the lowest standard deviation, the most uniform strain energy distribution. Figure 4.12 shows a visualization of one cell after having a completed the strain energy computation. The red struts have a large elastic strain energy and the blue ones have low elastic strain energy. Figure 4.13 is another visualization for a four cell trabecular architecture with the same coloring scheme.

After developing the trabecular stress calculator, it was packaged into the general selection function. The selection function takes in a list of genomes, points onto which those genomes map, and a set of initial conditions. It returns the genome that produces the best trabecular structure—as defined by lowest standard deviation in strain energy—for that stress state and those set of points.

The selection function begins by translation each genome and calculating the strain energy on all the struts in the system using the strut energy function. Next, the standard deviations in strain energy for each genome are calculated and compared. The system with

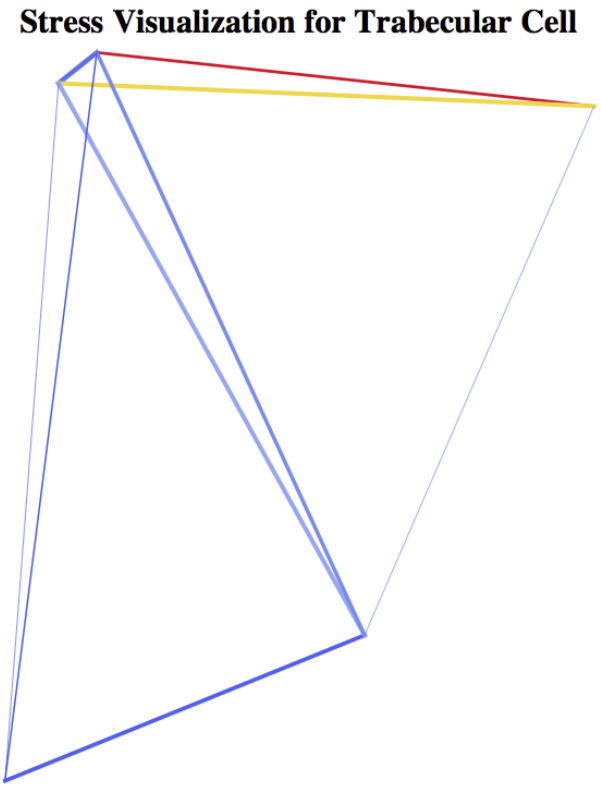

Figure 4.12: Visualization of the stresses on a strut in a cell in a trabecular system. Blue coloring indicates low stress, while red indicates a large stress

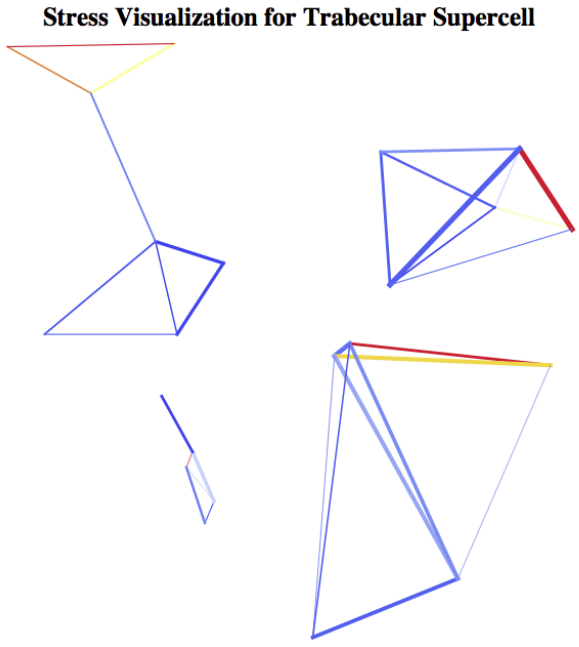

Figure 4.13: Visualization of the stresses in a two by two supercell.

the lowest standard deviation is chosen and it is selected by the function.

The output format is a list containing the selected genome, a translation of the genome into the developed trabecular system along with the strain energy of every strut in that system. Also output is the mean strain energy and standard deviation of the system. This output format allows for easy comparison and analysis.

The selection function is the where the power of a genetic algorithm is generated. By only propagating mutations that improve the mechanics of the system, it turns random mutations into a cumulative selection process, where only mutations that improve the performance of the structure propagate. Furthermore, it allows mutations to build on one another, producing a structure that is the product of incremental improvements.

To better interpret the results of the selection function, a stress visualizer function was developed that takes the output of the stress calculation and outputs graphical representations of the trabecular structure in such a way that the strain energy of all the struts can be compared. It does so using a colored scaling function, which colors the struts according to the ratio of that strut's energy to the maximum energy of all struts in that cell. Low energy struts are colored blue while high energy struts are red. Figures 4.12 and 4.13 were created using the stress visualizer function.

4.7 Reproduction

Reproduction is the part of the algorithm when a genome is input and copies of the genome with different mutations are generated. The output of the reproduction function depends on the mutation rate and the number of offspring per generation. The function determines if a mutation on a gene occurs by generating a random number between zero and one for that gene. If the mutation rate is greater than that number, that then that gene is mutated and its value will change. This can occur for any of the genes in the system: strut thickness,

angle, or length, and also the strut weight gene and the number of struts gene. The number of offspring that the reproduction function outputs is another variable that must be specified.

Because a trabecular system is a product of many individual trabecular cells, each with its own genome, there will always be mutations during reproduction unless the mutation rate is extremely small. Each cell has five different genes and whole systems have many cells. Therefore, the probability that at least one gene will be mutated during reproduction is large.

4.8 Non-Dimensionalizing

In order to make the results of algorithm more widely applicable, the lengths and stresses in the system were non-dimensionalized. As described in the selection selection, the strain energy calculation was normalized by the Young's Modulus. The lengths of the system were normalized by the length of the cell, the standard dimension for the problem.

The dimensionless strut length genes were defined in terms of fractional length of the struts relative to the size of the cell. The dimensionless thickness genes were defined in terms of the thickness to length ratio. The strut ranker function was adjusted to reflect these dimensionless parameters. The comparisons between the features of a possible strut and the ideal genome strut were done on the normalized features of the possible strut.

Once the strain energy was normalized, the only input that needed to be specified was the Poisson's ratio, which according to literature sources is 0.3 (Bagge 2000). The initial conditions that defined a stress state are dimensionless strains defined relative to the size of the overall cell. Non-dimensionalization allows for the comparison between systems of different sizes and makes the algorithm more broadly applicable.

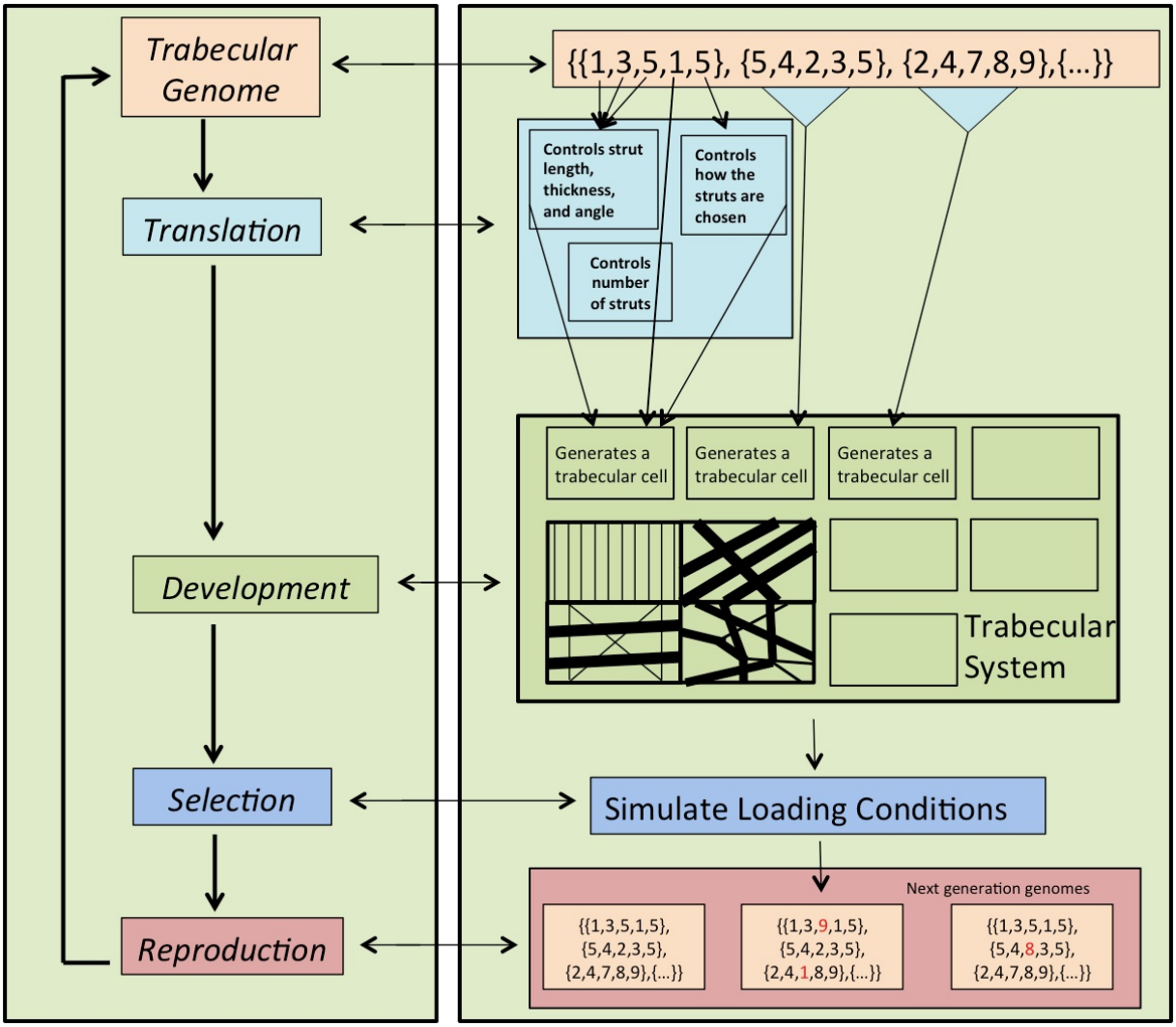

Figure 4.14: Schematic of trabecular microstructure evolution via genetic algorithm.

4.9 Evolution Function

An evolution function was created by merging the different development, selection, and reproduction functions together into a unified package. The input of evolution is an initial genome, set of points, and stress state, along with the number of generations of evolution, the mutation rate, and the number of offspring per generation. Figure 4.14 shows a schematic of the evolution process for this algorithm.

Evolution begins by calculating the strain energy of the initial genome, generation zero.

It then passes the results of that calculation to an evolver function, which replicates the genome, calculates the strain energies of all the offspring, and uses the selection function to output the best result. By running the evolver function with the Mathematica function NestList the output of each generation is used as the input for the next. The output of evolution is the genome, stressed translation, mean stress, and standard deviation for each output. Figure 4.15 shows the evolution of one test system across five generations using the stressed translation output. Figure 4.16 shows the standard deviation in strain energy as a function of the generation. Because of the random nature of mutations, the standard deviation can increase from one generation to the next even though the long term trend is to decrease.

Having now created and tested the trabecular genetic algorithm, the next step was running simulations to learn about both the performance of the algorithm and the microstructure evolution of trabecular systems under different loading conditions. This begins by first creating a set of test cases from which useful information can be extracted.

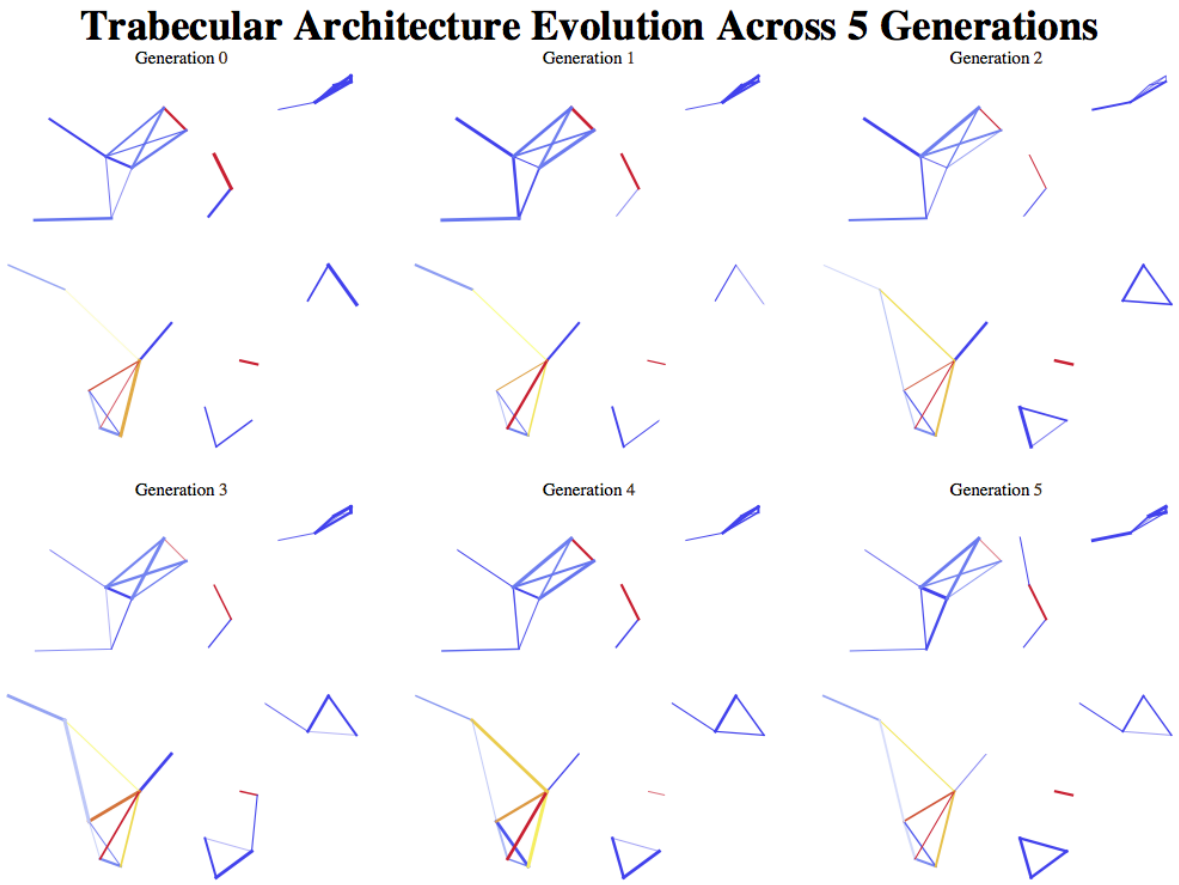

Figure 4.15: Evolution of a small test trabecular architecture with random loading conditions. The struts are colored according to their strain energy density, with red representing a strut with a large stored energy.

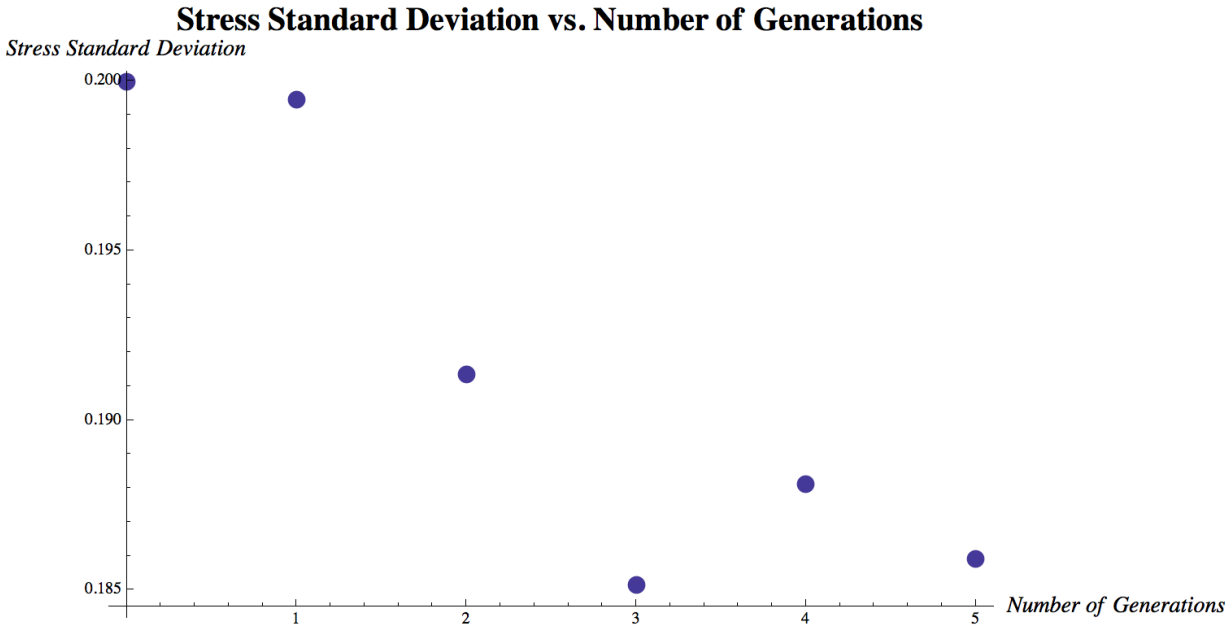

Figure 4.16: Stress standard deviation for the trabecular architecture from figure 4.14.

Chapter 5

Simulating Trabecular Microstructure Evolution

In order to investigate the microstructure evolution of trabecular bone, a series of simulations were created to demonstrate different aspects of genetic evolution. The simulations were designed to test how the evolution of a system depended on the genome, the points onto which the genome is mapped, and the loading conditions. The first set of simulations compared the evolution of two systems with identical sets of points and stress states but different genomes. The second set explored what happens to evolution of identical genomes and stress states but mapped onto a different sets of points. A third simulation looked at how different loading conditions affect the evolution of two identical genomes mapped onto the same set of points.

A second set of simulations were performed to explore how the performance of the algorithm depended on mutation rate, number of offspring per generation, range of genes, and number of generations. Because these results of this set of simulations focuses more on the algorithm and less on trabecular systems, they will be explored in more detail in the discussion section. Table 5.1 shows the different inputs that were varied for all the sets of simulations. The strain initial conditions were two percent compressive strain in either the

Num. of Offspring	Mutation Rate	Num. of Generations	Variable
4	0.2	10	Genome
4	0.2	10	Points
4	0.2	10	Stress
2, 4, 6, 8, 10	0.2	10	Offspring
4	0.2	5, 10, 15, 20, 25	Generations
4	0.05, 0.1, 0.15, 0.2, 0.25	10	Mutation Rate

Table 5.1: Table of parameters of trabecular evolution simulations

pure x or pure y direction uniformly distributed across the top and bottom of each cell. Each of the two sets of simulations was performed with two different ranges for the genes. In the first set, the gene controlling the number of possible vertices ranged from ten to twenty and the gene controlling the number of possible struts range from five to ten. The second set used between 100-200 possible vertices and 50-100 possible struts. The ranges of all the other genes were identical. In the large scale simulations, there were 5,354,808,750 different unique genomes that one cell could have, almost forty times more than the 135,877,500 possible genomes in the smaller case. All of the simulations were conducted on four cell systems, so the total number of possible genomes for a trabecular architecture is four times the number for an individual cell.

Because of both the random nature of mutations and the random nature of generating the points onto which a genome is matched, it would take a statistical analysis of multiple trials of each simulation to do quantitative analysis. Unfortunately, this was not possible due the computational demands of running such simulations, even using only four cells per trabecular system. However, it is still possible to extract meaningful trends from the set of results that were obtained.

In general, systems tended to converge to a minimum stress standard deviation within the first five generations, after which the genome continued to mutate but without changing the standard deviation. This indicates that there are multiple genomes which translate to a trabecular architecture well adapted to a specific set of loading conditions and points. At

times however, the undesirable combinations of mutations for those conditions did cause the standard deviation to increase.

5.1 Varying the Genome

The results of the first set of simulations show how the evolution process changes for two different genomes mapped onto the same set of points under the same stress. Although genome two was initially better adapted for the applied stress state, within a few generations genome one had reached the same level of performance. Figure 5.1 visualizes the systems at different points in the evolution process. Over the course of the evolution the average strut in each cell in the two systems change features considerably, adapting its length, thickness, and angle. Figure 5.2 shows the initial and final average struts for each cell in the two systems. The final average struts are different for the two systems and that indicates that structures with different macroscopic features can be equally well adapted for a given set of conditions.

5.2 Varying the Points

When testing the effect of varying the points while keeping the genome and stress state constant, it was discovered that applying the same genome to different points produced two systems with almost identical average initial struts, as defined by their angle, thickness, and length. Over the course of the evolution, the average struts changed and by the end of ten generations the length of the average strut of the first set of points was considerably shorter than the second set. Figure 5.3 shows the average struts for each cell before evolution in black and the final average struts in red. Interestingly, although the systems evolved differently, both reached the same level of performance. Figure 5.4 shows the initial and

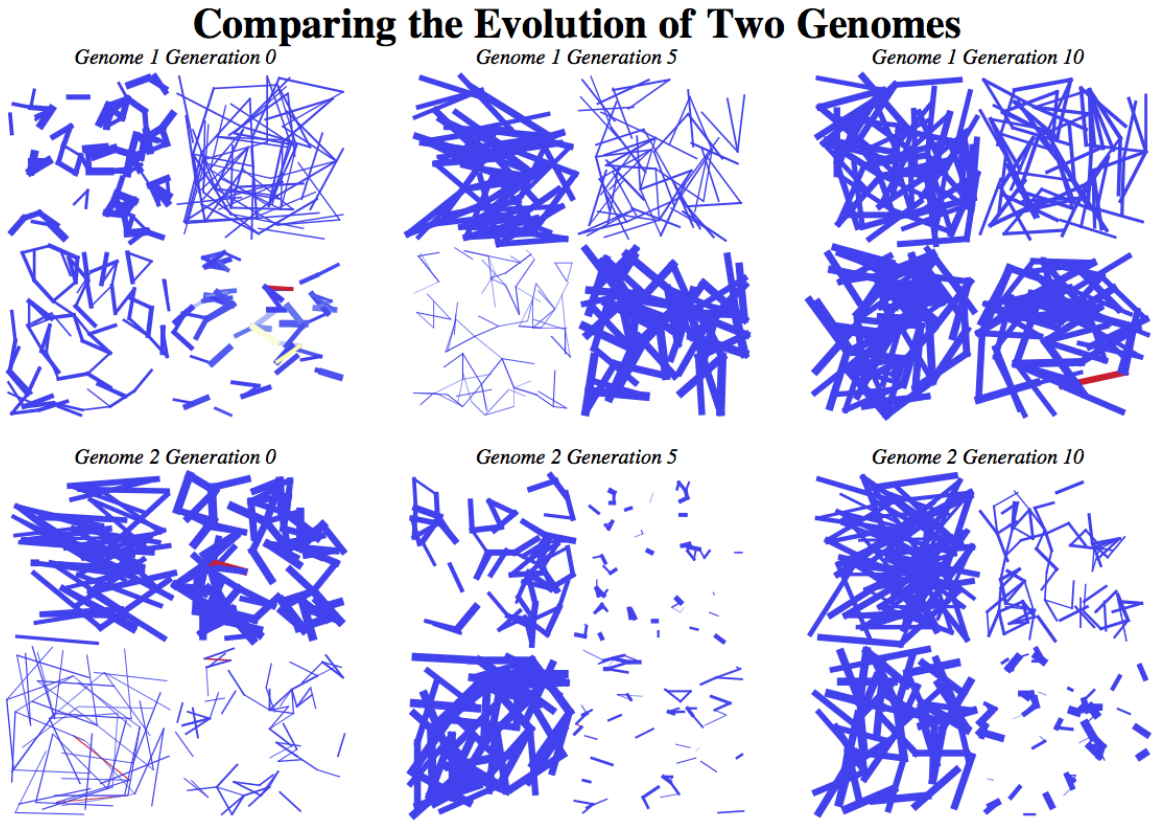

Figure 5.1: Trabecular Architectures for evolution for two different genomes mapped onto the same set of points under the same stress state at three stages of evolution. The figure shows how drastically the microstructure of the system can change in only a small number of generations and how the general tendency is for the number of struts with a large strain energy—bright red struts—to decrease.

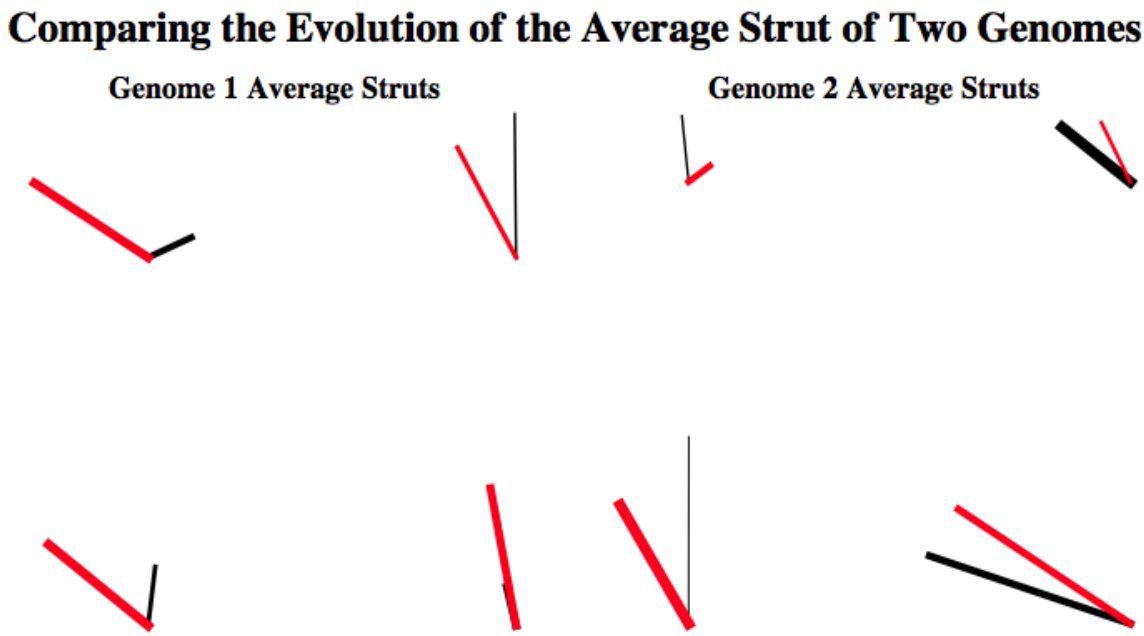

Figure 5.2: The black struts represent the initial average length, thickness, and orientation of each cell in the two systems. The red struts represent the average strut at the end of ten generations of evolution.

final architectures of the two different systems.

5.3 Mapping Genomes onto Points

After noticing how the average features of the cells of a given genome seemed invariant to the set of points onto which the genome was mapped, the average features of a genome mapped onto five different sets of points were calculated. The results are shown in figure 5.5. Although there was variation in the average angle with the points, the average length and thickness were essentially invariant with the points. The sensitivity of angle may be due to the weight gene, which may have discounted the weight of keeping the angle constant relative to the other two genes. The invariant nature of the average features of the cell shows how the genome can be thought of as encoding macroscopic scale instructions about average features within a system which can then be multiply realized on different sets of points.

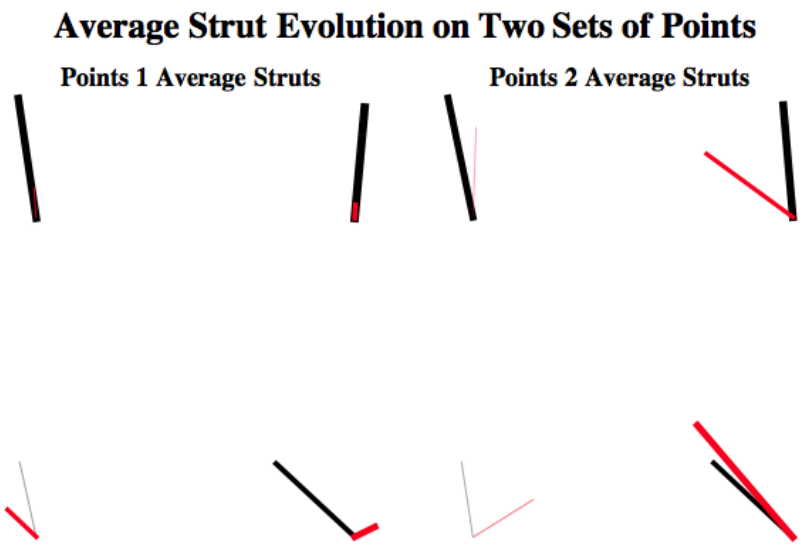

Figure 5.3: The average initial and final struts in each cell in the two systems shown in black and red respectively.

5.4 Varying the Applied Stress

The first stress state featured a compressive load of 2 percent of the initial cell size in the y direction. The second stress state featured the same 2 percent load but in the x direction. Both cases used the same initial genome and the same set of lattice points. This meant that the initial trabecular architectures were identical. As expected though, the different loading conditions caused the systems to evolve with different features. Figure 5.6 shows the architecture of the two systems initially and after five and ten generations of evolution.

5.5 Decrease in Standard Deviation

The metric of performance for the algorithm was the standard deviation in stored strain energy in the struts. A lower standard deviation in stored strain energy means that all the struts are under equal stress and are being used equally to support the load, indicating an optimized architecture. Because the smaller scale simulations had fewer struts and more

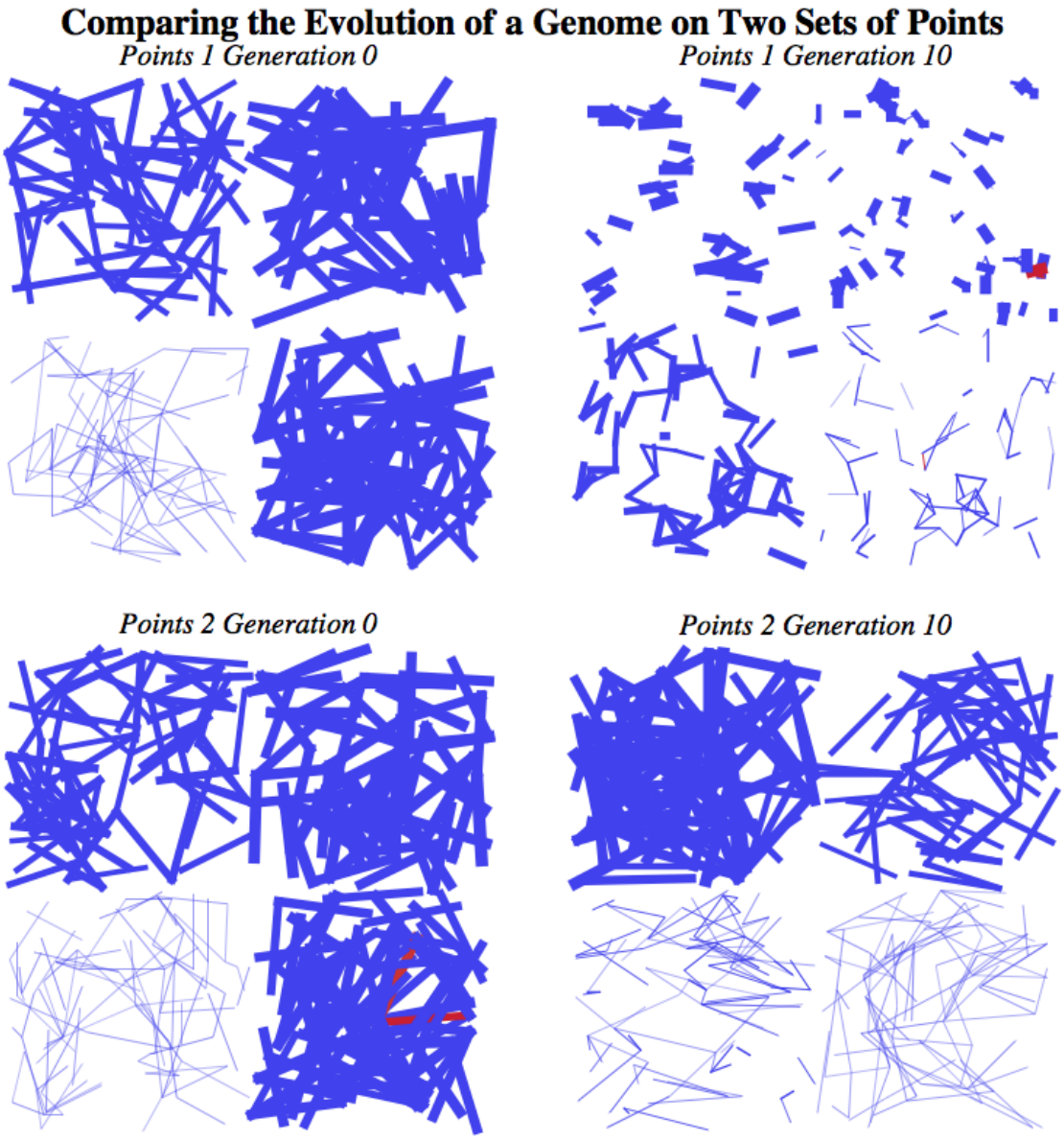

Figure 5.4: The initial and final architectures of two identical genomes mapped onto two different set of points both under the same stress state.

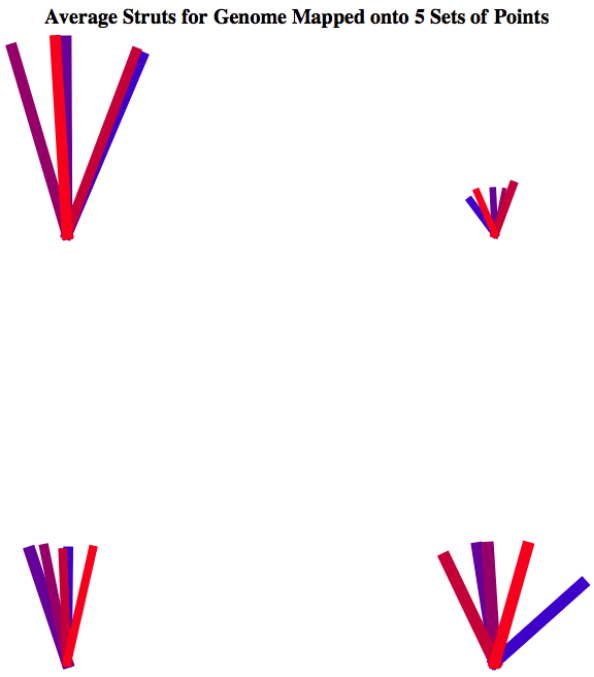

Figure 5.5: The average struts for a single genome mapped onto five different sets of points. The length and thickness of the average strut seem to be invariant to the set of points onto which they are mapped, but the average angle can vary.

struts were initially stressed, there was larger initial standard deviation and improvements were larger. Figure 5.7 shows the standard deviation in strain energy divided by the initial standard deviation for the small scale case of two different genomes. Both genomes show a decrease in standard deviation of more than 90 percent.

5.6 General Trends

Overall, the results of the simulations demonstrate that the genetic algorithm is able to use cumulative selection to improve the performance—as defined in terms of the standard deviation in strain energy—of the microstructure of trabecular bone in response to a given set of loading conditions. The ability of different systems to reach the same level of performance indicates that there are multiple genomes which generate well adapted structures.

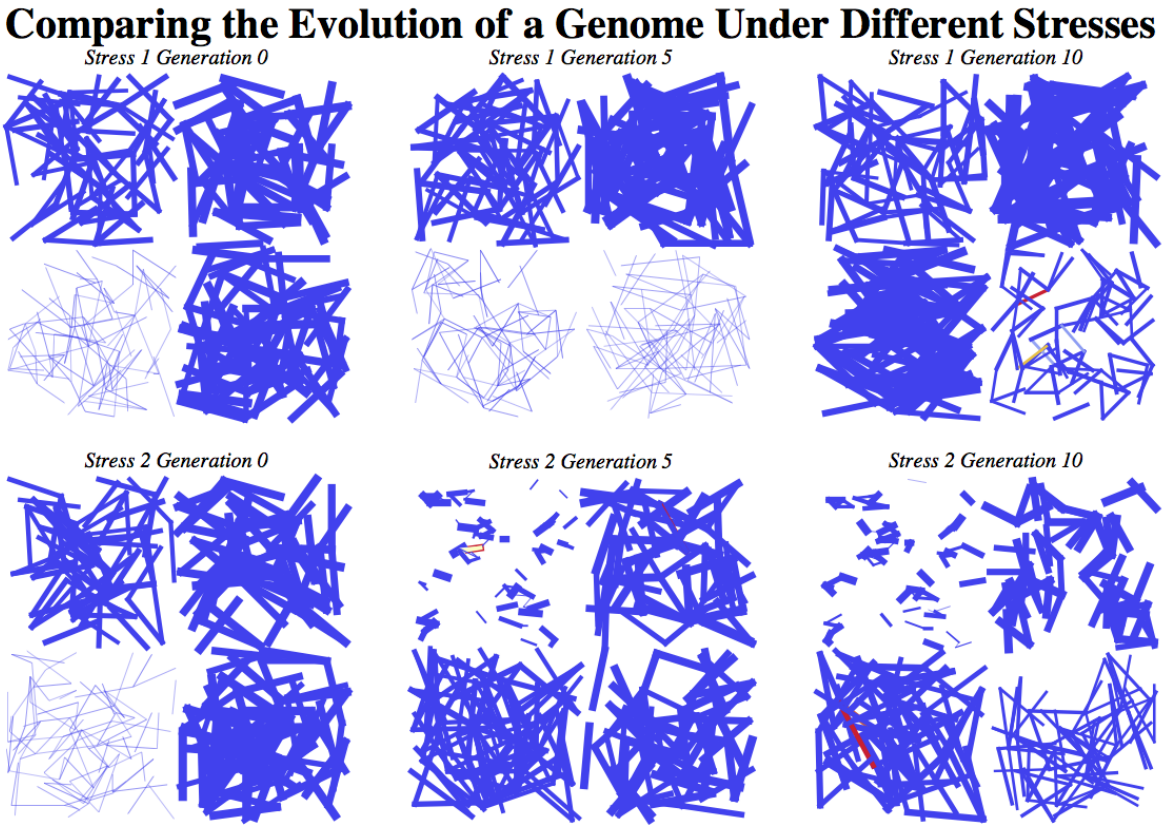

Figure 5.6: Showing how the average features of a genome vary with the points onto which it is mapped. The average angle appears to be more sensitive to changes in mapping points than the thickness or length.

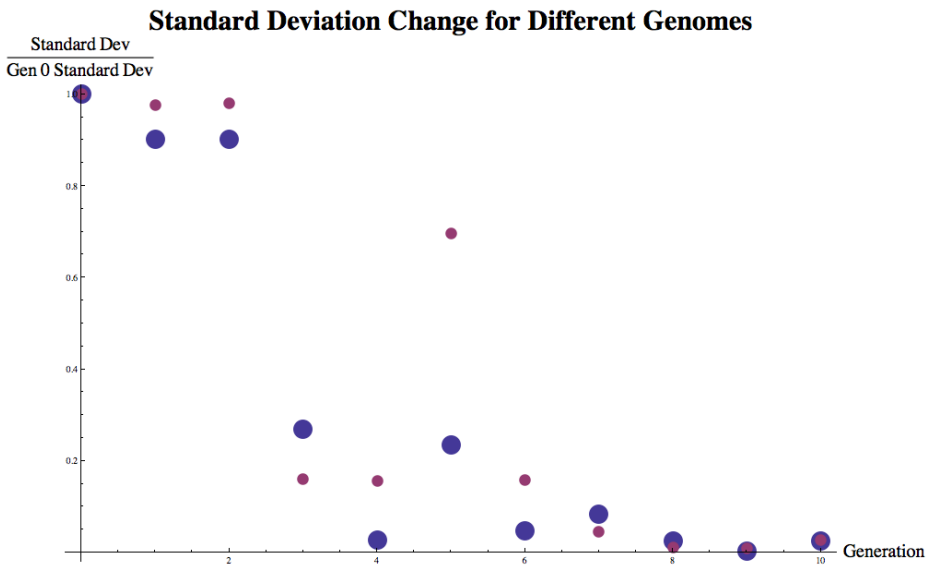

Figure 5.7: Graph of the standard deviation in strain energy divided by the generation zero standard deviation shows how the system improves with increasing number of generations. The larger blue dots are for the first genome, while the smaller red dots are for the second genome.

The occasional decrease in performance in subsequent generations shows imperfect nature of cumulative selection; that sometimes the wrong combination of mutations results in a structure that is not as well suited as the previous generation. Repeating the simulations in order to generate enough results to preform statistical analysis should allow for more quantitative measures of the observed trends.

Chapter 6

Discussion

The set of simulations investigating the effect of mutation rate, number of generations, and number of offspring per generation were designed to explore what governs the ability of the algorithm to generate optimal structures. Performing these simulations on both the large and small scale systems demonstrated how the range of genes affects performance as well. In addition to exploring how those factors influence the overall algorithm, this section will also discuss the benefits and limitations of the model, compare the genetic algorithm to other methods of modeling the optimization of trabecular microstructure. Potential future work will also be explored, including the possibility of extending the algorithm to 3D systems.

In the large scale simulations, it was observed that the systems tended to very quickly converge to a standard deviation and then maintained that deviation even as the genome continued to mutate. The robustness of those systems may be accounted for by the fact that the number of struts that were stress in the large scale system was small relative to the overall number of struts. As a consequence, the system only needed to change a few struts to reduce the standard deviation without concern for how it affected the rest of the system. In the future, applying more initial conditions will result in more struts being stressed and more incremental improvements with each subsequent generation.

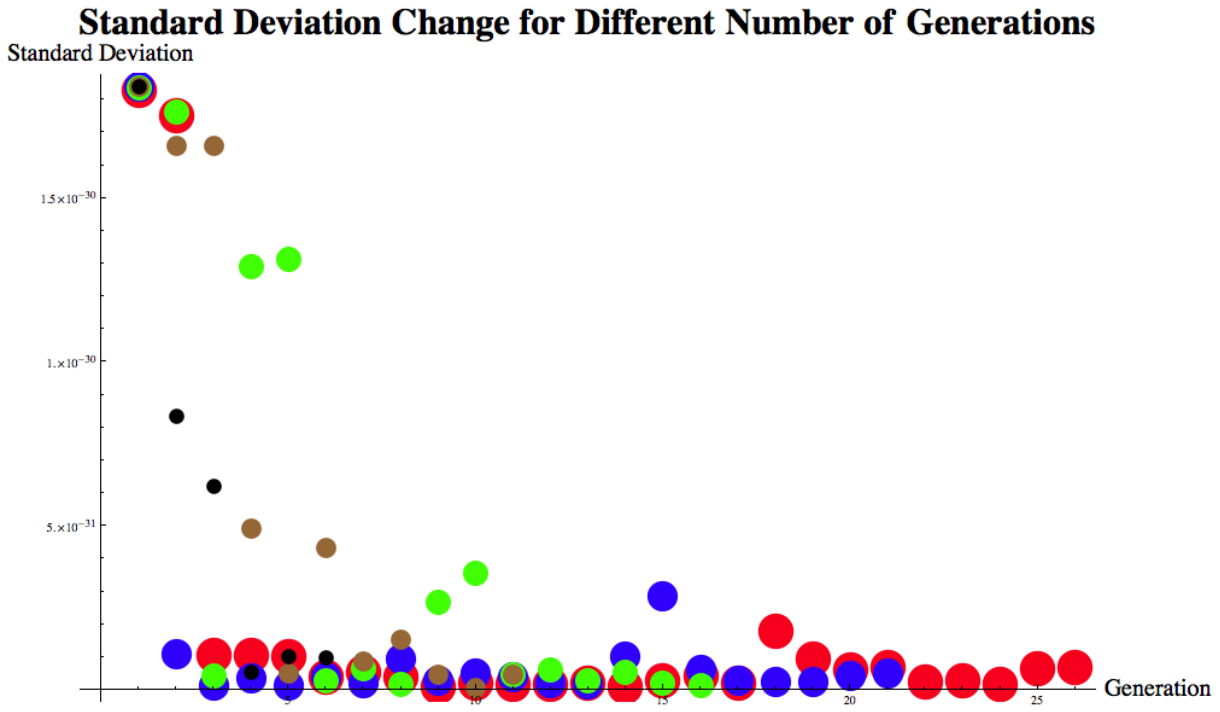

Figure 6.1: The graph of standard deviation for different numbers of generations shows the effect of convergence: once the system finds the minimum standard deviation additional generations of evolution have little effect on performance. The largest dots, in red, are for the system that under went 25 generations of evolution. Decreasing dot size, from blue to green, to brown, to black, corresponds to the different number of generations of evolution the different systems underwent.

6.1 Number of Generations

Although one might expect that systems that underwent more generation of evolution would result in a lower overall standard deviation, what was observed instead was convergence around a minimum value, usually within five generations, followed by oscillations around that minimum. Figure 6.1 shows a plot of the standard deviation as a function of generation for systems with five different numbers of generations. Each color dot represents the evolution of a single system and the dot size increases with the number of generations of evolution. The graph shows that all the systems tended to converge around the same value for the standard deviation, some systems converged faster than others. Figure 6.2 shows the same plot but

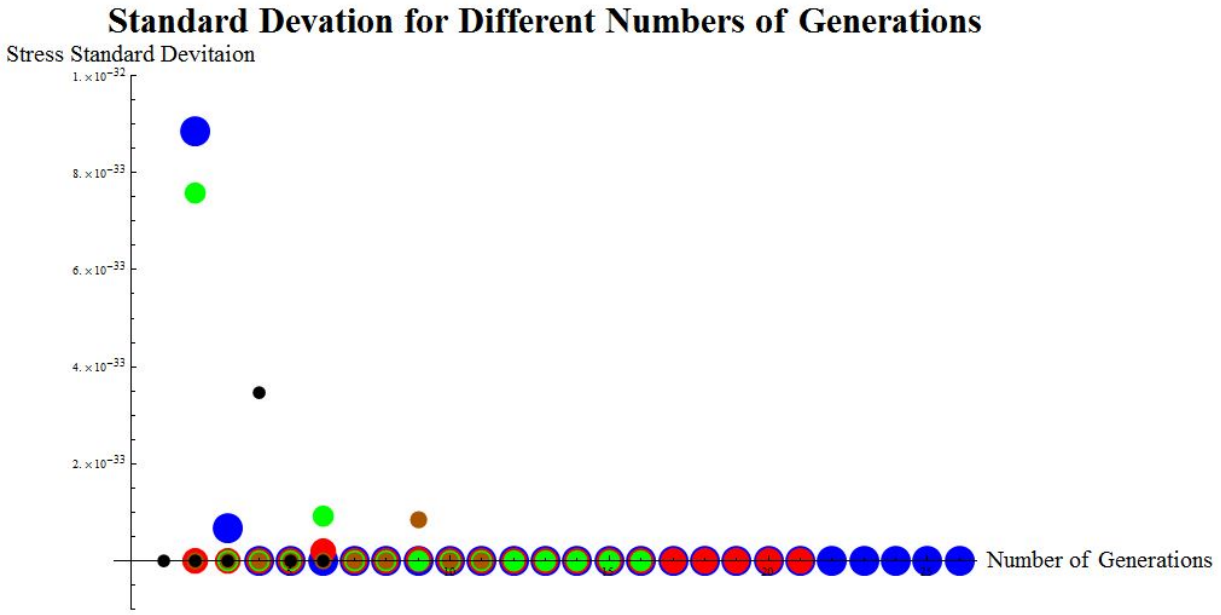

Figure 6.2: The graph of Standard Deviation for the larger systems shows how the trend of convergence is even more pronounced. Again, the dot size is proportional to the number of generations of evolution the different systems underwent.

for the large scale simulations. It shows completely stable convergence and no additional improvements in performance beyond generation five, indicating that the system has most likely reached the best possible performance level for that set of conditions.

6.2 Number of Offspring

Increasing the number of offspring per generation increases the likelihood that only mutations that improve performance will propagate. This is reflected in a lower variance in the standard deviation once the system is conserved. Figures 6.3 and 6.4 demonstrate this trend plotting the standard deviation in strain energy after convergence against mutation rate for both the large scale and small scale systems. The large scale algorithm showed a larger increase in stability than the small scale; the large scale case with ten offspring per generation had a variability in converged strain energy with additional generations nine orders of magnitude

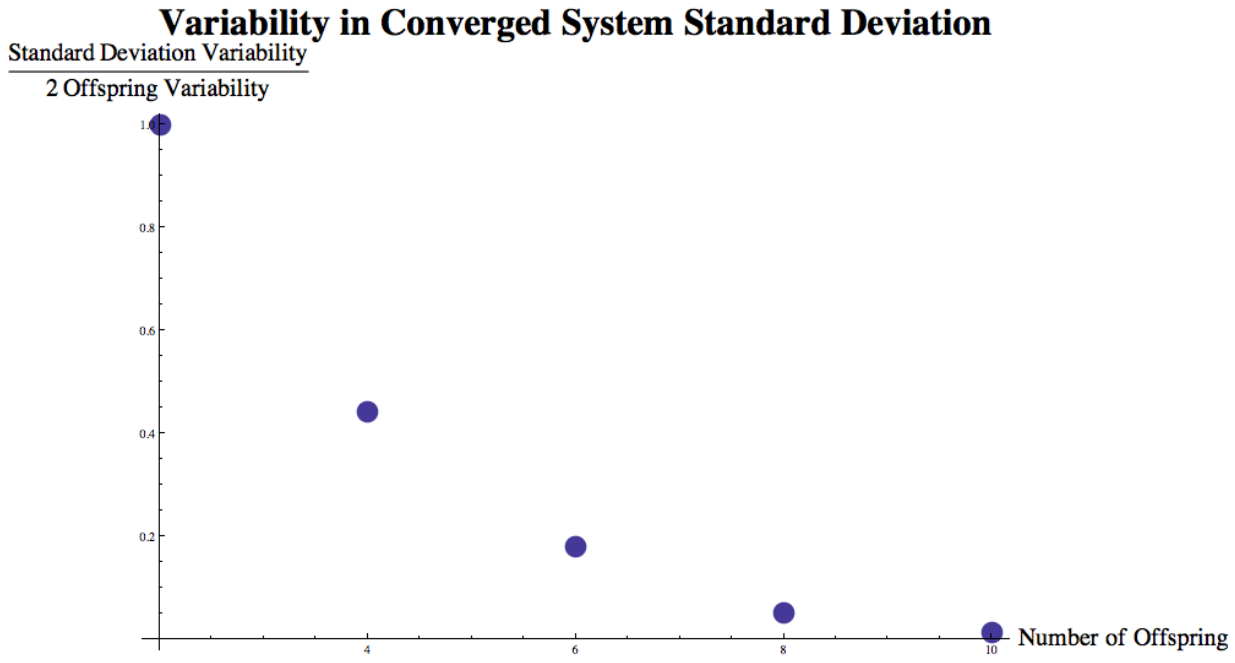

Figure 6.3: Increasing the number of offspring makes the system's performance level more stable and reduces the variability in standard deviation with subsequent generations

lower than the case with only two offspring per generation.

Although increasing the number of offspring per generation improves the stability of the system, it comes at the expensive of computational speed. Each additional generation requires its own stress minimization calculation, the most demanding computation in the algorithm. In the large scale set, for example, the case with eight offspring per generation took three times as long to simulate as the case with only two offspring per generation.

6.3 Mutation Rate

Mutations are the reason that cumulative selection generates such remarkable results, and the mutation rate greatly affects the performance of the algorithm, as already noted from the radiating line algorithm. A higher mutation rate means a higher probability of getting a large increase in performance in the first few generations. The flip side of this is that a higher

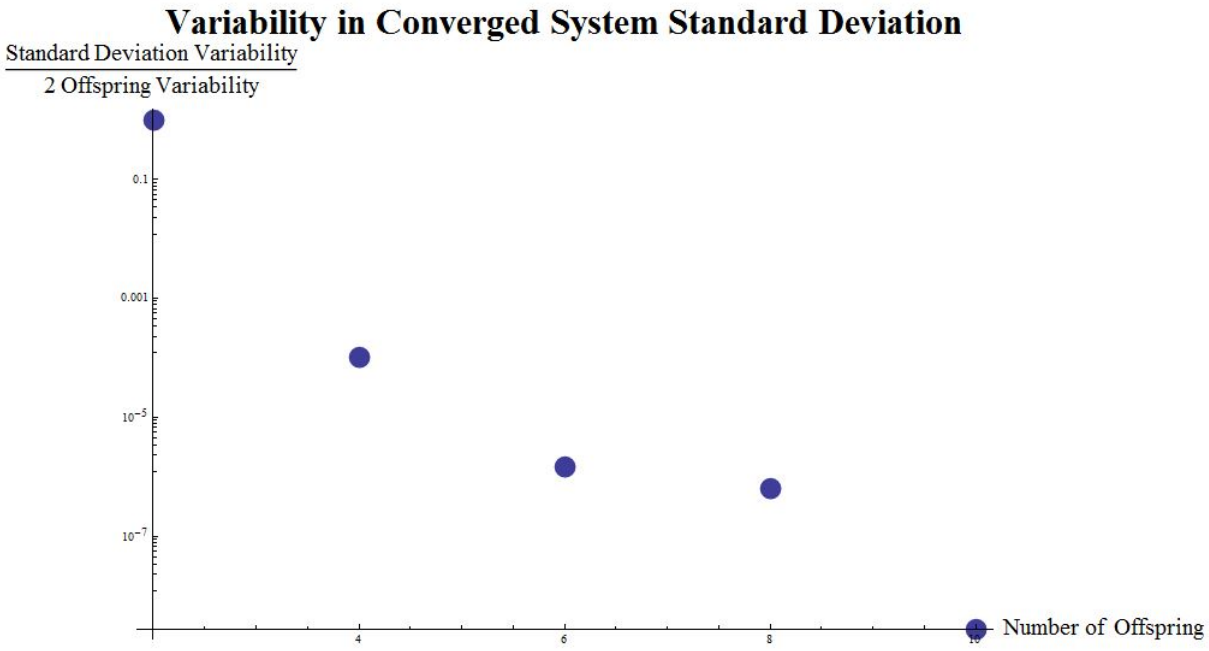

Figure 6.4: For the larger system, the improvements in stability are even more pronounced; the trial with 10 offspring per generation had variability nine orders of magnitude lower than the trial with only two offspring, as shown on this log plot.

mutation rate results in a less stable converged system because of a higher probability of the genome mutating away from a well-adapted form. However, the large scale system exhibited almost no variability once converged, even with mutation rates as high as 50 percent. Figure 6.5 shows the standard deviation against number of generations for all the different mutation rates. This implies that there are so many potential genomes that produce a well-adapted system for those conditions that mutations on multiple offspring per generation will likely result in one with a nearly identical standard deviation. This contrasts with the radiating line genetic algorithm, which had only 729 possible genomes, and where increasing the mutation rate resulted in a strong increased in variability. This supports the hypothesis that having a large number of genomes reduces the potential negative effects of a large mutation rate.

Also supporting that hypothesis is the fact that the smaller scale simulations did show that a higher mutation rate was more likely to generate variance in the converged standard

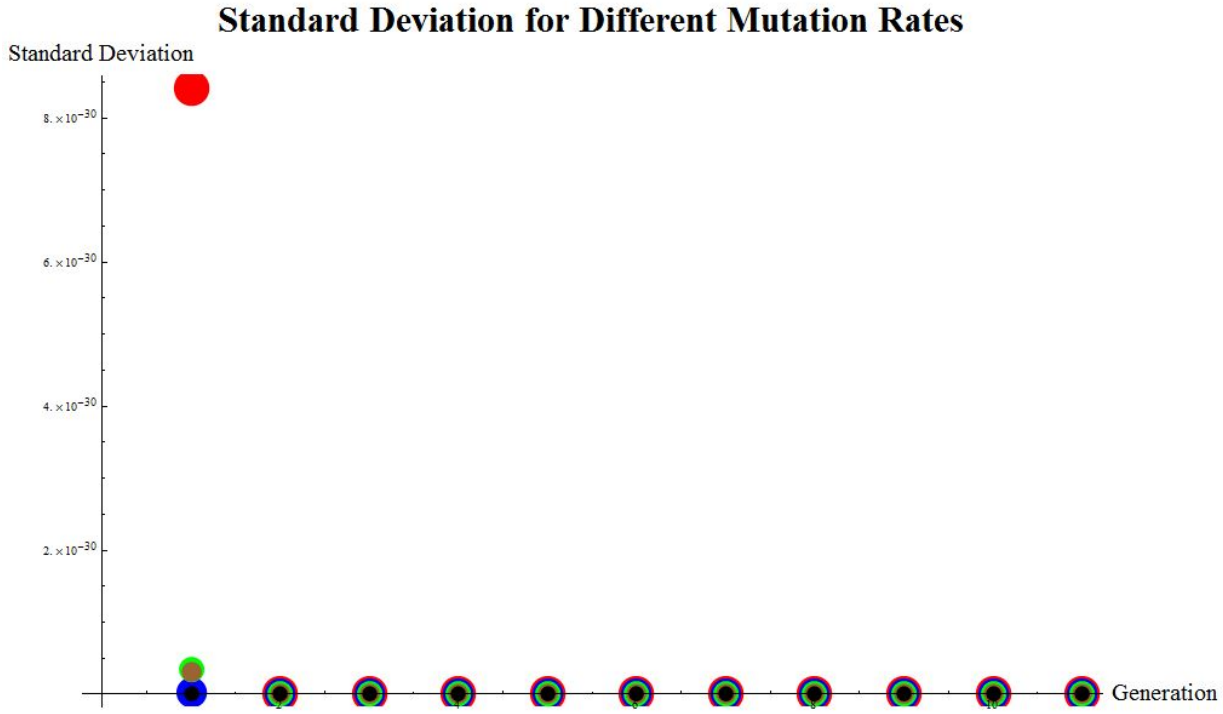

Figure 6.5: In the large scale system, increasing the mutation rate did not greatly affected variability, presumable because the number of mutations that produced a system with equivalent performance was large enough that one was always produced. This time, the point size scales with mutation rate, with the largest mutation rate corresponding to the set of red points and the smallest corresponding to the black.

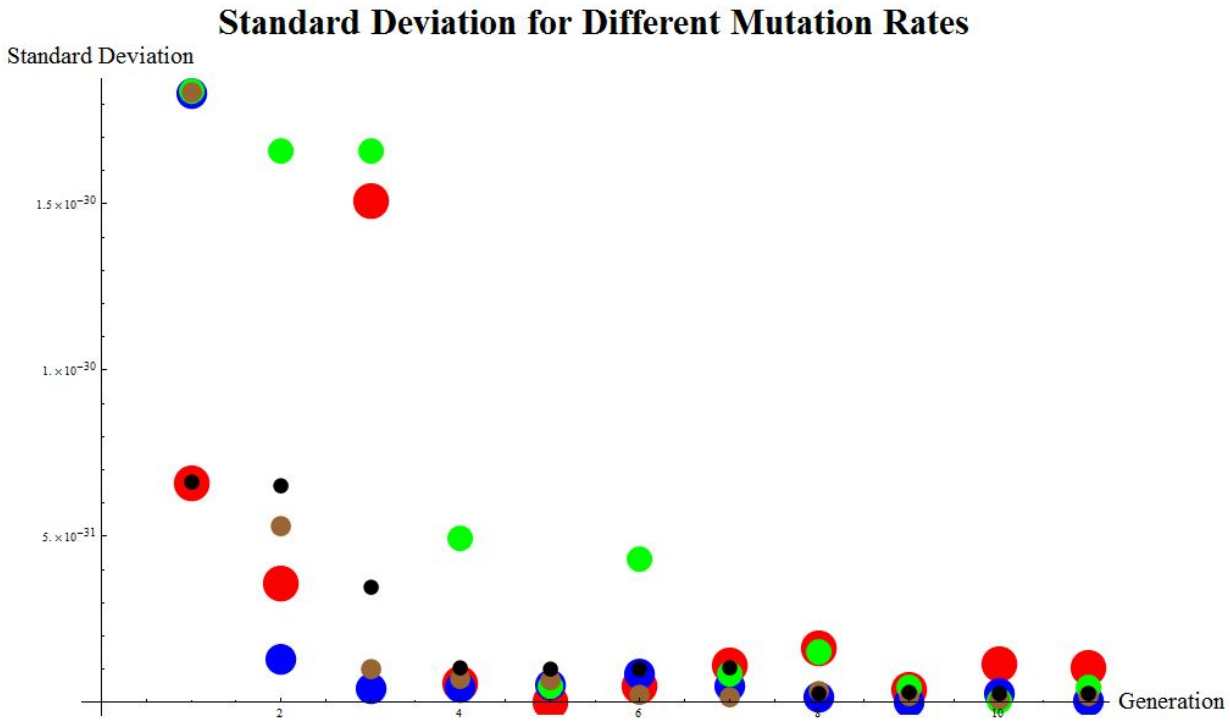

Figure 6.6: The small scale system showed that increasing the mutation rate did have an effect on system stability. Again, the mutation rate scales with the point size—from red to blue, to green, to brown, and to black

deviation. Figure 6.6 shows the standard deviation against number of generations and figure 6.7 shows the variance in converged standard deviation as a function of mutation rate. The trend in figure 6.7 is increased variance with increased mutation rate, although a statistical analysis is necessary to confirm this observation. Overall, varying the mutation rate allows for a balance between fast initial improvements and making improvements stable. Furthermore, because changing the mutation rate does not change the expense of the simulation, it is arguably the most important parameter in genetic algorithms.

6.4 Rate Limiting Computation

As the number of offspring, number of generations, and size of the system varied, so did the time it took to complete the simulations. The results can be used to determine how

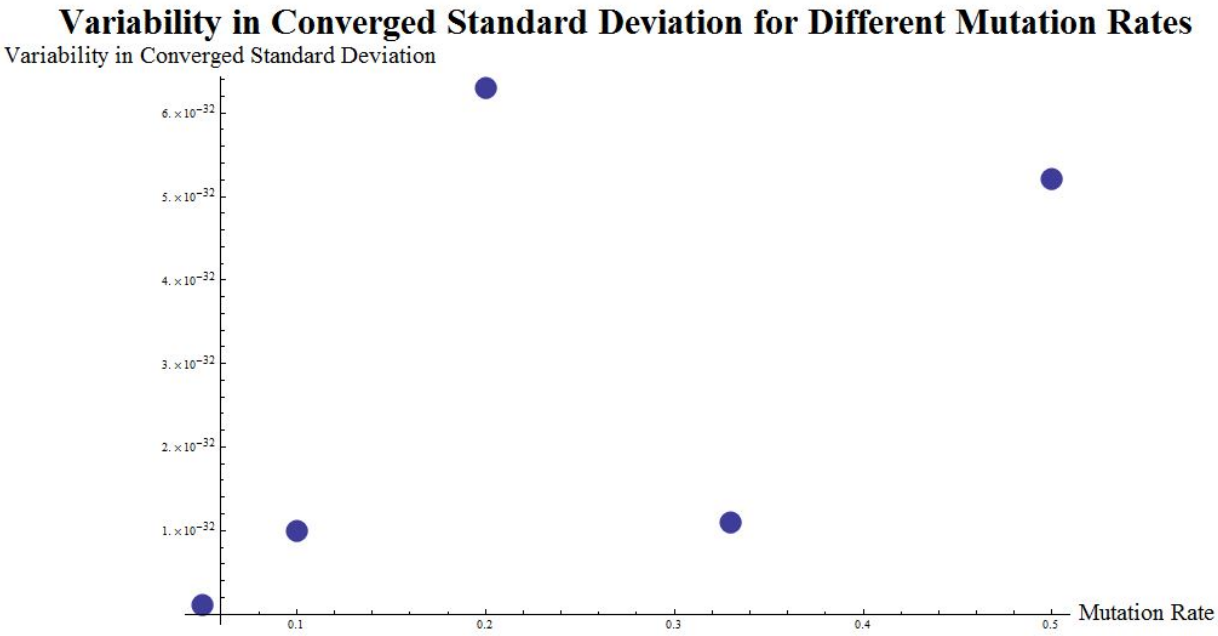

Figure 6.7: The simulations with a larger mutation rate tended to have a larger variability in converged stress than those with a smaller mutation rate.

sensitive the time it takes to complete a computation is to those parameters and what the rate limiting step is. The length of the computation scales linearly with the size of the system, as shown by a 100 fold increase in time per simulation when moving from the small scale to large system, where the large scale system had 10 times more vertices and 10 times more struts. Within a size, the system is most sensitive to the number of offspring. Figure 6.8 shows a plot of the length of computation time for both the case of varying the number of offspring and the number of generations. The slope of the offspring curve was 2.4 times steeper indicating that the length of the computation is most sensitive to this parameter.

6.5 Limitations of Genetic Algorithms

The main limitation of using a genetic algorithm to evolve a system from an initial state to an optimized state is the dependence on the genome and the set of genes. In order for the model to be useful, the genes must accurately reflect the degrees of freedom of the system and

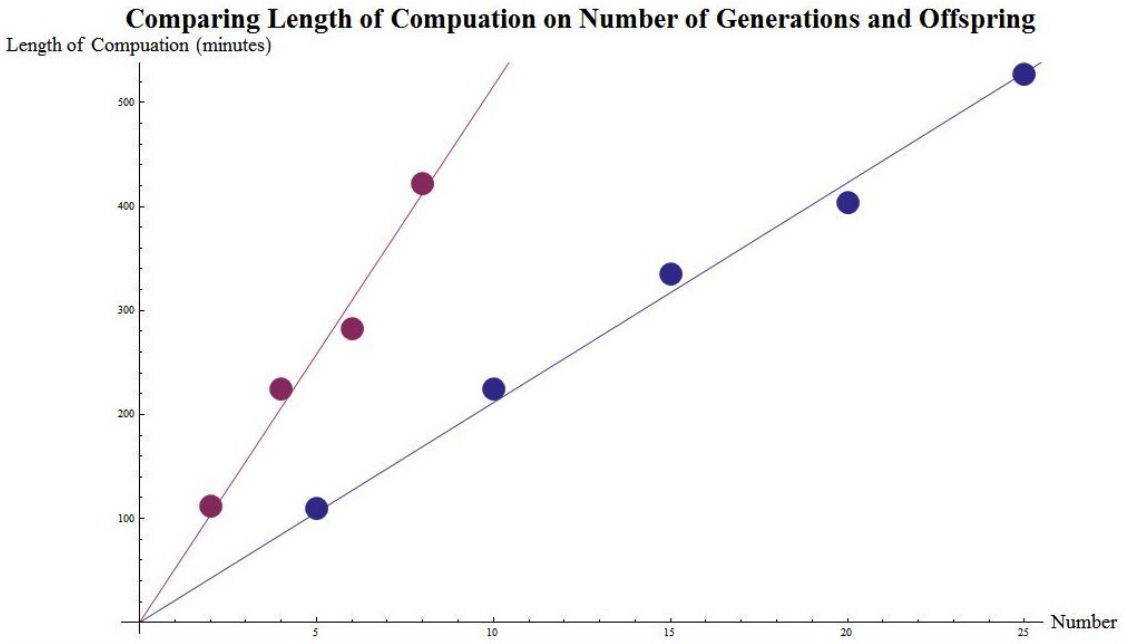

Figure 6.8: The plot shows in red the increase in length of computation with varying number of offspring per generation while the blue dots are for varying number of generations. The steeper slope of the red line indicates that time is more sensitive to the number of offspring.

also must have the correct range of values. Without the correct set of genes, it is impossible to accurately model the evolution of a system and what to parameters are most important for the desired set of properties.

Another limitation is the importance of selection criterion. The key feature of the cumulative selection process is a quantitative comparison of systems according to some predefined selection criterion. If the wrong selection criterion is used, then the system will not evolve toward the state with the desired properties. Another limitation of this type of model is that it does not capture the role of driving forces in the evolution process and how they are responsible for the adaptation of different regions of the trabecular systems.

A limitation of this method specific to modeling trabecular microstructure evolution is that a completely new set of trabecular struts is created after each generation of evolution. In contrast, actual trabecular adaption occurs via adding and removing material to the surface of existing struts. Therefore, this algorithm is better suited for modeling the process of

microstructure evolution across generations of a species—because each generation has a new set of struts—rather the microstructure evolution of the preexisting architecture within a specific organism.

6.6 Comparison with Previous Work

The previous work which had been done on modeling microstructure evolution in trabecular systems focused on the process by which macroscopic properties like stress are used to determine whether a given strut grows or shrinks. Previous models also focused on the adaptation of one structure. In contrast, the genetic algorithm generated new structures every generation and operated by choosing the best possible structure of all the ones in a given generation.

Another area of distinction between this model and previous work is that other models had no predefined stopping point built into them while the genetic algorithm runs for a predetermined number of generations. The studies that used a remodeling rate, for example, have built into their model that the system is stable when the remodeling rate on all struts is zero. There is no way of knowing if a system is stable using a genetic algorithm and the evolution process will always generate new structures with each subsequent generation, even if those structures are not ultimately selected. While the optimal structure was defined in previous cases as the structure where the remodeling rate on all struts is zero, there is no predefined optimal structure in a genetic algorithm. Instead, it is more of a continuous search process.

Like previous models, the genetic algorithm was able to model the process of trabecular microstructure evolution and was able to reduce the standard deviation of the strut strain energy—indicating a optimally designed fully stressed structure. A distinction is that the struts in previous models adapted by adding or removing pixels to their surface, while the

genetic algorithm creates a completely new set of struts each generation. Both methods, however, proved able to model the adaptation of a trabecular architecture to a set of loading conditions.

6.7 Future Work

There are a number of interesting possible considerations for future modeling work using the genetic algorithm. Thus far, the cases and simulations run have validated genetic algorithms both as a method for modeling trabecular microstructure evolution and as a method for generating optimal structures for a given stress state. This work also began to unpack how the performance of a genetic algorithm depends on parameters like the mutation rate, number of generations, and number of offspring. While this work yielded important insights on both genetic algorithms and trabecular microstructure, it also raised the possibility of a number of follow up studies and simulations.

6.7.1 Larger Scale Trabecular Simulations

Because of limiting computing resources, no simulations were preformed on systems larger than four cells. A full scale trabecular architecture, however, consists of many more cells. Running simulations on larger scale systems, like the 64 cells in figure 6.9, is the most logical next step of this work because it will reveal the process of adaptation at the full trabecular scale. In addition, these larger scale simulations should also allow for the emergence of domains within the overall architecture.

In addition to simulations on larger systems, another logical next step is to add more initial conditions and simulate more complex stress states. Simulation the loading conditions that a human experiences upon running and comparing that to the loading conditions of a dog running is one possible interesting example. Adding more initial conditions will result

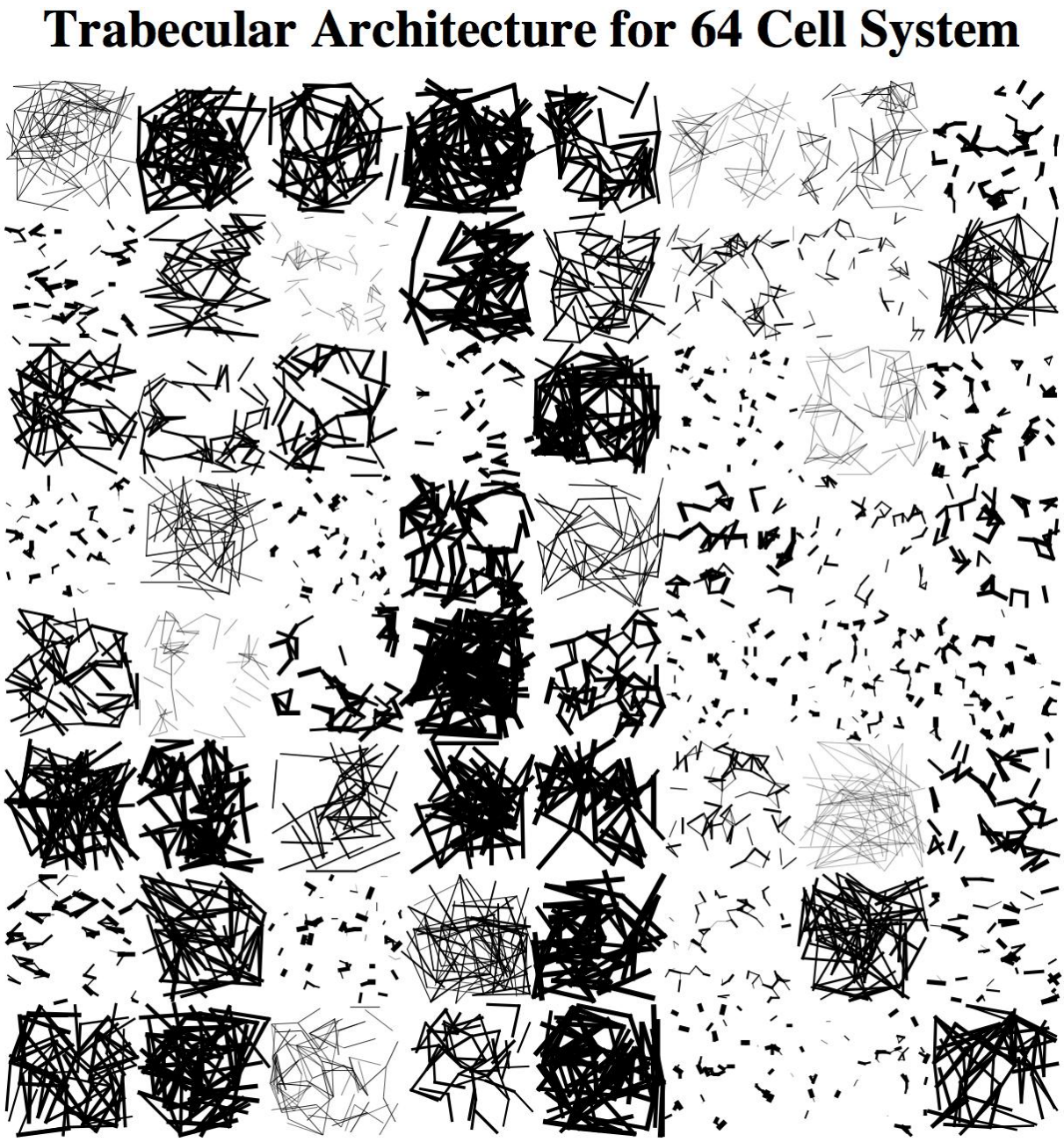

Figure 6.9: Trabecular architecture for a 64 cell system.

in more fine tuned adaptation because more struts in the system will be under an initial stress and be forced to adapt to reach a uniform stress distribution. It is important though, that these future simulations be run multiple times so that the results can be averaged and a statical analysis can be conducted.

6.7.2 Changing the Trabecular Genome

Another possible way in which to extend this work is to change the features of the trabecular genome. Adding genes, like one for a chemical potential of calcium or one to adjust the trabecular strut density, means additional degrees of freedom in the trabecular architecture and a more responsive model. On the other hand, changing the types of genes can result in an algorithm which more accurate models the process of microstructure evolution. Changing the genes may also prove a way to combine genetic algorithms with the previous modeling efforts.

In actual trabecular bone, rather than encoding for the features of struts, the genes control the sensitivity and rate of the osteoclasts and osteoblasts. Creating a trabecular genome with these features would result in system which evolves by adding and removing material to the surface of preexisting struts, instead of generating a new set of struts each generation. Changes in the genome would then result in changes in the sensitivity of the system—and thus changes in when material is added or removed to a strut—and also changes in the rate of remodeling or the ratio of osteoclast to osteoblast cycle time. These changes would result in a genetic algorithm which more accurately models the process of trabecular microstructure evolution.

Whether adding additional genes related to the strut ranking criterion the current algorithm uses or designing a new set of genes, expanding the genome increases the degrees of freedom of the system and provides more phase space in which to fine tune the optimal structure. The methodology for development is flexible enough that it can be adapted to

any type of genome of arbitrary dimension. Extending the genome to create a more accurate model is another natural extension for this work.

6.7.3 Extension to 3D

A second possible place to take this work is to extend the algorithm to 3D. Because the trabecular are modeled using graph theory, this is very easy to do. It only requires mapping connections onto points in 3D space instead of 2D space. Then the struts can be ranked according to the same procedure as 2D, but with an additional gene for the second angle. Of course, extending this work to 3D will make the stress minimization calculation much more complex and make the algorithm more computational expensive, but it is interesting to explore the types of 3D structures that can be produced.

6.7.4 3D Printing and Experimental Validation

3D printing offers another potential avenue in which to take this research. 3D printing offers a chance to bring into existence the structures the algorithm generates, to test the structures under actual loading conditions, to explore their mechanical properties, and to attempt to confirm the increases in performance that the algorithm predicts. One possible experiment which can be preformed is to 3D print the a trabecular structure with a genome designed for a specific stress state and to test its performance against a trabecular structure with a random genome. Anther possible experiment is to compare the performance of an initial system and system that has evolved for a given stress state. This experiment provides a way to test experimentally whether or not the structures that algorithm generates do in fact become better adapted to its environment.

6.8 Genetic Algorithms for Materials Optimization

One important take away from both the trabecular microstructure genetic algorithm and the radiating line genetic algorithm is the flexibility of this modeling method. Genetic algorithms can be used across materials science and engineering—and indeed all of science and engineering—in order to generate structures optimized for a given set of functions or properties. Furthermore, the framework established in this thesis provides a method for generalized genetic algorithm development, independent of the system being modeled.

The first step to creating a genetic algorithm for any materials science application is to determine how the characteristics of a system can be described using a genome. For an algorithm modeling polymer properties, the genes may be the average chain length, the side group, and tacticity, for example. The next step is to create a procedure for development—the process of translating the genome into a structure or system. For an algorithm modeling microstructure of metals, the development may require using the theory of kinetics to determine how the genes of phase fraction, temperature, and cooling rate determine the microstructure. Following the development is the selection phase, which requires identifying the metric of comparison for the system. In the case of metal microstructure, the selection criteria may be highest tensile yield strength. Once the selection criterion is established, it is necessary to quantify a procedure for calculating the criterion using the structure that results from the development phase. The last step in generic algorithm development is the reproduction phase, where the best genome is reproduced and mutated so that the cycle of evolution may be repeated.

What should be clear is the features of genetic algorithms—development, selection, and reproduction—can be mapped onto any possible materials optimization problem. As long as there is known criterion for the optimal properties and a clear relationship between said property and the structure of material, a genetic algorithm can be used to find the best

structure for a set of given conditions, just as it was for trabecular structures under a given load.

Chapter 7

Conclusion

The ability for *in vivo* adaptation is one property that separates biomaterials from their inorganic counterparts. Biomaterials like trabecular bone have the ability to adjust their structure in response to changes in their environment. A genetic algorithm is one way to model that process of adaptation, using the mechanism of cumulative selection through which biomaterials evolved their unique properties. For materials scientists and engineers seeking to developing the next generation of materials, biomaterials offer important lessons about the relationship between structure and function and how adaptability can create robust structures even in the face of large uncertainties about the environment in which they will be used.

This thesis modeled trabecular microstructure evolution using a genetic genetic algorithm. After creating the algorithm, simulations were conducted using different parameters to explore microstructure evolution using the process of mutation and cumulative selection. The results of the simulations show that the algorithm is able to increase the performance of the system as defined by the selection criterion. The results showed the tendency of the systems to converge to a minimum standard deviation in strain energy, with subsequent variability of the converged value inversely proportional to the number of offspring per gen-

eration and proportional to the mutation rate. The results also show how a many different possible architectures can be optimal for a given set of loading conditions and how small changes in a genome or even a single gene can completely reshape the overall architecture of the system.

By successfully modeling the process of trabecular microstructure evolution, this work also establishes genetic algorithms and cumulative selection as a methodology for the optimization of material's properties. For materials scientists seeking to optimize the properties of synthetic materials, genetic algorithms offer a method for creating a diverse set of structures and systematically searching through those structures for the ones that result in the best properties. Whether trying to optimize the chemistry and molecular weight of a polymer for tissue engineering, the microstructure and grain boundaries of a metal for an industrial application, or the electrical properties of a semiconductor, genetic algorithms use the accumulation of beneficial mutations to improve properties until they reach a given performance metric. For materials scientists and engineers designing the next generation materials, genetic algorithms serve as an important tool as they attempt to synthesis materials with structures, functions, properties, and performance better than even the best biomaterials.

Chapter 8

References

Adachi, T., Tomita, Y., Hiroshi, S., Masao, T., 1997. Simulation of Trabecular Surface Remodeling based on Local Stress Nonuniformity. The Japan Society of Mechnical Engineers Internal Journal 40, 782-792.

Bagge, M., 2000. A model of bone adaptation as an optimization process. Journal of Biomechanics 33, 1349-1357.

Boyle, C., Kim, I., 2011. Three-dimensional micro-level computational study of Wolff's law via trabecular bone remodeling in the human proximal femur using design space topology optimization. Journal of Biomechanics 44, 935-942.

Carter, D. Beaupre, G., 2001. Skeletal Form and Function. Cambridge, UK, pp.138-158.

Cotter M., Loomis D., Simpson S., Latimer B., Hernandez C., 2011. Human Evolution and Osteoporosis-Related Spinal Fractures. PLoS ONE 6. 26658. doi:10.1371/journal.pone.0026658.

Cranford, S., Buhler, M. 2012. Biomateriomics. Springer, New York, pp. 27-60

Dawkins, R., 1996. The Blind Watchmaker. W. W. Norton and Company, New York, pp. 43-76.

Galik, K., Senut, B., Pickford, M., Gommery, D., Treil, J., Kuperavage, A., Eckhardt, R., 2004. External and Internal Morphology of the BAR 100200 Orrorin tugenensis Femur. Science 305, 1450-1453.

Gibson, L., 1985. The Mechanical Behaviour of Cancelllous Bone. Journal of Biomechanics 18, 317-328.

Huiskes, R., Riumerman, R., Harry van Lengthe, G., Janssen, J., 2000. Effects of mechanical forces on maintenance and adaptation of form in trabecular bone. Nature 405,

704-706.

Jang, I., Kim, I., 2008. Computational study of Wolffs law with trabecular architecture in the human proximal femur using topology optimization. Journal of Biomechanics 41, 2353-2361.

Kemp, T., Bachus, K., Nairn, J., Carrier, D., 2005. Functional trade-offs in the limb bones of dogs selected for running versus fighting. The Journal of Experimental Biology 208. 3475-3482.

Keten, S., Xu, Z., Ihle, B., Buehler, M., 2010. Nanoconfinement controls stiffness, strength and mechanical toughness of beta-sheet crystals in silk. Nature Materials 9, 357-367.

Pontzer, H., Lieverman, D., Momin, E., Devlin, M., Polk, J., Hallgrmsson, B., Cooper, D., 2006. Trabecular bone in the bird knee responds with high sensitivity to changes in load orientation. The Journal of Experimental Biology 209, 57-65.

Rook, L., Bondioli, L., Khler, M., Moy-Sol, S., Macchiarelli, R., 1999. Oreopithecus was a bipedal ape after all: Evidence from the iliac cancellous architecture. Proceedings of the National Academy of Science 96, 8795-8799.

Tsubota, K., Suzuki, Y., Yamada, T., Hojo, M., Makinouchi, A., Adachi, T., 2009. Computational simulation of trabecular remodeling in human proximal femur using large-scale voxel FE models: Approach to understanding Wolffs law. Journal of Biomechanics 42, 1088-1094

Turner, C., 1992. On Wolffs Law of Trabecular Architecture. Journal of Biomechanics 25. 1-9.

Watson, J., Crick, F., 1953. Molecular Structure of Nucleic Acids. Nature 4356, 737-738.

Wolff, J., 1892. Das Gesetz der Transformation der Knochen. Hirchwild , Berlin.