A Tensegrity Based Structure Optimization Framework

by

Vyacheslav Iourivich Jdanov

A thesis submitted to the

Graduate Program in Computing

in conformity with the requirements

for the degree of Master of Science.

Queen's University

Kingston, Ontario, Canada

August 2014

Vyacheslav Iourivich Jdanov August 25, 2014

Abstract

This thesis presents a tensegrity based structure optimization framework. A tensegrity structure consists of rods (compressed elements) and elastics (tensile elements) connected into a stable structure. A tensegrity structure diffuses the forces applied to any one of its elements to the entire network of connected elements, making the whole of the structure flexible and strong.

This thesis focuses on structural optimization problems that are bound in the physical world and subject to real-world laws; examples include optimization of a bridge design and optimization of an electrical grid. The framework presented in this thesis exploits the properties of tensegrity networks as an efficient, powerful, low-level heuristic to steer the optimization. The tensegrity modeling framework stipulates that a tensegrity model consists of three components: an optimization goal (specifies the method by which the performance of an instantiation of the tensegrity model is evaluated), a graph specification (defines the properties of a tensegrity graph, including edge labels and attributes that can appear in this graph), and an adaptation rule set (specifies the change over time in the attributes of a tensegrity graph).

A tensegrity graph provides a versatile platform for structural modeling and optimization, because attributed nodes and edges can be used to satisfactorily represent the essential properties of a large variety of optimization problems. This is illustrated in the three examples: a detailed and instantiated model of fascia, a conceptual bridge model, and a model abstracting the intricacies of an electrical grid. The components of each model are very easy to modify, allowing for rapid prototyping and insight into the significance of the variables altered between prototypes. The framework is generic, allowing for models of varying degrees of abstraction while remaining intuitive and applicable to the structural optimization problems they represent.

Acknowledgements

I would like to thank Dr. Dorothea Blostein for her supervision and guidance throughout this work, without her support my thoughts would be coherent to no one but myself.

Contents

Glossary

Acceleration	-	A change to the Velocity Vector of a Node, caused by Internal Forces and External Forces acting on the Node.
Adaptation Rule Set	-	A Component of a Tensegrity Model. This Component specifies the change over time in the Attribute values of a Tensegrity Graph.
Attribute	-	An Edge attribute or a Node attribute in a Tensegrity Graph. The set of Attribute values defines the State of a Tensegrity Graph.
Component	-	One of the three components of a Tensegrity Model: Optimization Goal, Graph Specification and Adaptation Rule Set.
Current Strain	-	An optional Edge Attribute. In the model presented in Section 3, this Attribute represents how much Strain the Edge is under at this time.
Edge	-	A connection between 2 Nodes (terminology from graph theory). Edges are labeled using the set of Edge labels in the Graph Specification. All the examples in this thesis use the labels *Elastic* and *Rod*. Every Edge must have an Attribute called Natural Length. An Edge applies an Internal Force to its Nodes.
Elastic	-	An Edge label; an Edge labeled *Elastic* pulls two Nodes together.

External Force	-	A force applied to a Node by an external cause. The causes are model specific, and used for purposes such as modeling gravity, wind or collisions. The External Force accelerates the Node according to F=mA; using m=1.
Graph Specification	-	A Component of a Tensegrity Model. This Component specifies properties of the Tensegrity Graph, including a specification of the Attributes and Edge labels that can appear in the Tensegrity Graph.
Integrity	-	An optional Edge Attribute. In the model presented in Section 2.1.2, this attribute represents how much Strain the Edge can withstand before breaking.
Internal Force	-	A force applied to a Node by one of its incident Edges. This accelerates the Node according to F=mA; using m=1.
Natural Length	-	A required Edge Attribute. This Attribute represents the length at which this Edge applies zero Internal Force to its Nodes. (The length of an Edge is the Euclidian distance between the Node Positions of its Nodes.)
Node	-	A connection point between Edges (terminology from graph theory). Every Node has two required Attributes: Node Position and Velocity Vector.
Node Position	-	A required Node Attribute. This Attribute represents the Node's current position in N-Dimensional simulation space.
Optimization Goal	-	A Component of a Tensegrity Model. This Component specifies the method by which the performance of an instantiation of the Tensegrity Model is evaluated. As discussed in Section 2.1.1, the Optimization Goal can be formulated as an activity audit, a fitness function, or performance benchmarking.

Resting State	-	An optional Attribute for Elastic-labeled Edges. This Attribute represents the amount of Strain that an Elastic is under in Structure Equilibrium.
Rod	-	An Edge label; an Edge labeled *Rod* pushes two Nodes apart.
State of a Tensegrity Graph	-	A set of values for the Attributes in a Tensegrity Graph.
Stiffness	-	A required Attribute for Elastic-labeled Edges. The Internal Force that an Elastic applies to its two Nodes is proportional to the value of the Stiffness Attribute.
Strain	-	A measure of the difference between the length of an Edge and its Natural Length. (Current Length - Natural Length)/Natural Length
Structure Equilibrium	-	A State of a Tensegrity Graph when all the Nodes' Velocity Vectors are zero (with constant External Force).
Tensegrity Graph	-	An undirected, attributed, connected, edge-labeled graph representing a tensegrity structure. (This is standard terminology from graph theory; a graph consists of Nodes and Edges, each with Attributes.)
Tensegrity Model	-	A model consisting of three Components: Optimization Goal, Graph Specification and Adaptation Rule Set. Instantiating a Tensegrity Model produces a Tensegrity Graph that is in accordance with the Graph Specification Component of this Tensegrity Model. The instantiation is done according to the guidelines provided by the Tensegrity Modeling Framework.

Tensegrity Modeling Framework	-	A set of guidelines and assumptions that support creation and instantiation of Tensegrity Models.
Velocity Vector	-	A required Node Attribute. This Attribute represents the velocity of the Node, measured as a sum of Internal and External Forces applied to it.

Chapter 1

Introduction

This thesis presents a tensegrity based structure optimization framework. This chapter describes the class of targeted optimization problems (section 1.1), explains tensegrity (section 1.2), provides an overview of the framework and associated terminology (section 1.3), and introduces the optimization problems used as examples throughout this thesis (section 1.4). Finally, section 1.5 lists the contributions made by this thesis.

1.1 Optimization problems

Optimization problems arise in many situations. A fitness function is used to grade how well a proposed solution solves the problem. The user analyzes the goal of the problem to define the fitness function and determines the variables that the fitness function will evaluate. The variables under consideration form the optimization problem's search space, which encompasses every combination of variables being evaluated as a potential solution. The optimal solution is the lowest point in the search space, representing the combination of variables to which the fitness function attributes the best grade. Each variable is a dimension of the search space and as the number of considered variables increases, the potential combinations increase exponentially. A larger search space increases the computation time required to solve the problem, making it necessary for the user to weigh the importance of the variable's input with the resulting computation cost.

Variables can be coupled with other variables. *Loosely coupled* variables have values that are nearly independent of one another. For example, in an optimization problem involving a vehicle transporting a liquid, the density of the liquid is loosely coupled with the speed of the vehicle. *Tightly coupled* variables have values that are highly dependent on one another, making certain combinations of variable values invalid because they do not form a viable solution. The coupling between variables affects the shape of the search space, increasing the importance of thorough navigation of the search space in order to find the optimal solution.

Various computational approaches for solving optimization problems are in common use [1]. *Exact Algorithms* are guaranteed to find an optimal solution, but in many cases are computationally expensive because they require a comprehensive view of the search space to avoid reporting a local minimum instead of a global one. *Approximation Algorithms* are guaranteed to find a solution that is within a specified factor of optimal, sacrificing a measure of accuracy to save on computation time. For many optimization problems it is very difficult to construct an efficient algorithm that provably finds a solution with a guaranteed proximity to optimality. Many optimization problems are known to be NP-hard. The best known algorithms for these problems are exponential in nature. Thus it is necessary to resort to heuristic algorithms. *Heuristic Algorithms* execute quickly, aiming to find the best solution they can in limited time, but with no guarantees about how close the solution is to optimal. The limitation of accuracy versus speed is particularly crippling in problems whose variables are tightly coupled.

This thesis focuses on structural optimization problems (described more thoroughly in Section 4.4), a subset of tightly coupled optimization problems bound in the physical world and subject to real-world laws. A vivid example of a structural optimization problem is the design of a bridge. The bridge must be robust, with the bridge optimized to tolerate numerous deviations in the values of operational parameters; failing to account for every combination of variable values may lead to the bridge's collapse. Similar, but more abstract structural optimization problems include traffic grids and electrical grids. The accuracy of optimization is doubly important for these problems, because suboptimal implementations can be very costly; in the case of a traffic grid, failure may precipitate a traffic jam, while in the case of a bridge, failure may result in a collapsed bridge. The framework presented in this thesis exploits the properties of tensegrity networks (discussed in Section 1.2) as an efficient, powerful, low-level heuristic to steer the optimization. This thesis aims to demonstrate that while this approach does not guarantee an optimal solution, it does guarantee a solution that is functional for the conditions it has been tested against.

1.2 Tensegrity

A tensegrity structure, as illustrated in Fig 1.1, consists of *rods* (compressed elements) and *elastics* (tensile elements) connected to one another. The connection points between rods and elastics are *nodes*. In a static tensegrity structure in equilibrium, the nodes are pushed apart by the rods and pulled together by the elastics. The result is a stable structure, which diffuses the forces applied to any one of its elements to the entire network of connected elements [4], making the whole of the structure flexible and strong [17].

1.3 The Tensegrity Modeling Framework

This thesis presents a Tensegrity Modeling Framework: a set of definitions and guidelines that support creation and instantiation of Tensegrity Models, and their use in solving optimization problems. These definitions and guidelines are explained in detail in Chapter 2. Technical terms, written with capital letters, are summarized in the glossary. Briefly, a Tensegrity

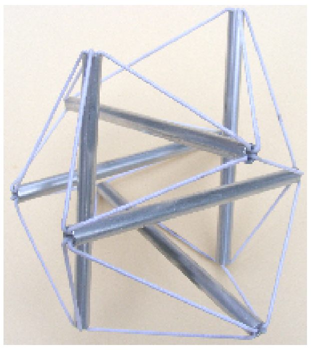

Figure 1.1: Example of an Icosahedron Tensegrity Structure. Reproduced from www.tensegrityinbiology.co.uk with permission: copyright by G. Scarr, 2006

Model consists of three Components: an Optimization Goal, a Graph Specification and an Adaptation Rule Set.

- The Optimization Goal specifies the method by which the performance of an instantiation of the Tensegrity Model is evaluated.
- The Graph Specification defines the properties of the Tensegrity Graph, including the Edge labels and Attributes that can appear in the Tensegrity Graph.
- The Adaptation Rule Set specifies the change over time in the Attribute values of a Tensegrity Graph.

Instantiation of a Tensegrity Model produces a Tensegrity Graph that is in accordance with the Graph Specification Component of the Tensegrity Model.

A Tensegrity Graph provides a versatile platform for structural modeling and optimization, because attributed Nodes and Edges can be used to satisfactorily represent the essential properties of a large variety of optimization problems. Edge attributes can be freely used to represent arbitrary properties of Rods and Elastics. Every Edge has a required attribute called Natural Length and can optionally have attributes such as Current Strain and Integrity (used in the example model in Chapter 3). Edges are labeled as either rods or elastics, and additional attributes may be defined for these subcases. In examples presented in this thesis, elastics have optional attributes including Stiffness (governs the Internal Force that the edge applies to its two nodes) as well as Resting State (represents the amount of Strain that the elastic withstands in Structure Equilibrium). Similarly, Node attributes can be freely used to represent properties of nodes; the two required node attributes are Node Position and Velocity Vector.

In a Tensegrity Graph, Edges exert Internal Force on their Nodes. Additional External Forces can be defined for purposes such as modeling gravity, wind or collisions. The combination of Internal and External Forces governs the simulated movements of the Tensegrity Graph. As described in section 4.1, the simulation proceeds by computing the Velocity Vector of each Node (the sum of Accelerations caused by Internal and External Forces acting on the Node) and adjusting the Nodes' positions accordingly. Because the Internal Forces are tied to the positions of the Nodes, the Velocity Vectors gradually decrease as the Tensegrity Graph approaches Structure Equilibrium. The Structure Equilibrium state is a solution to the optimization problem.

1.4 Example optimization problems

Section 4.1 presents the technical steps for instantiating optimization problems as models on the Tensegrity Modeling Framework. The framework exploits the flexibility and stability of tensegrity structures to provide unsupervised structure optimization to any problem that can be instantiated as a Tensegrity Model. The following examples are presented:

- Chapter 3 presents a detailed model of a Fascial Network in 2D space. Fascia (connective tissue) adapts its structural properties based on the local forces it is subjected to.

- Section 4.2 explores the process of optimizing a bridge structure for greater endurance under the expected operating conditions. By simulating weather and traffic patterns for the area, the bridge can be optimized for that particular location.
- Section 4.3 explores the optimization of a more abstract structure - an electrical grid. By simulating historical strain data, the causes of overloads and blackouts can be determined and by simulating hypothetical situations other flaws can be detected before they become problems.

1.5 List of contributions

This thesis presents a novel framework for the unsupervised optimization of tensegrity structures (Section 2.1), guidelines for instantiating optimization problems as models on the framework (Section 2.2), steps for simulating the resulting models (Section 4.1), a detailed example model (Chapter 3) and a number of hypothetical example models (Sections 4.2 and 4.3). Finally Chapter 5 discusses other avenues for exploration, and summarizes the evidence that the contributions have the following desirable properties:

The Structure Optimization Framework (Chapter 2)

- generic - can be used to model a wide variety of physical systems
- applicable - can be used to create effective abstractions of physical systems, representing only those aspects of the system that are required for optimization
- intuitive - clear and straightforward method of instantiating optimization problems as models on the framework
- useful - faster to define system and adaptation rules in framework than to use other methods (neural networks, finite element modeling, etc)

Usability of the Framework (Chapters 3 & 4)

- interactive - can be modified mid-simulation to examine resulting behavioral changes
- rapid prototyping - quick to alter configuration and learning rules
- insight - can run multiple simulations in parallel to better illustrate the differences between parameters

Model of Fascia Optimization (Chapter 3)

- fidelity - can simulate distributed structure of myofascia and musculoskeletal systems to explore models of homeostatic mechanisms at work in biological systems
- rapid prototyping - quick to alter configuration and learning rules
- insight - can simulate situations with various adaptation rules to determine which ones most closely match observed responses of fascia
- structural memory - can offer insight into the scaling of adaptation rates for structural memory

Alternative to Neural Networks for Studying Emergent Properties (Section 3.5)

- structural memory - the adaptive tensegrity system has a memory based on optimizing structure according to encountered input. This contrasts with neural network memory oriented towards pattern recognition and information retrieval
- graceful degradation - can be used to measure overall performance loss from local injury and determine injury thresholds for catastrophic failure
- continuous learning - adaptation algorithms can use adaptation history to bias future adaptation, simulating real world body/brain learning
- undirected activation - all components of a model are points for input

Chapter 2

Tensegrity Based Structure Optimization Framework

This chapter presents the tensegrity based structure optimization framework. Section 2.1 defines the framework's components, while Section 2.2 covers the steps necessary to utilize those components and instantiate a model.

2.1 Components of the Framework

Technical terms, written with capital letters, are summarized in the glossary.

2.1.1 A - Optimization Goal

The Optimization Goal defines how the performance of the Model is evaluated. This information is used during model instantiation to guide the formulation of Attributes and the Adaptation Rule Set. Several methods can be used to measure a Model's optimization; the choice of method depends on the complexity of the goal.

1. Activity Audit - a measure of the rate of change within a Tensegrity Graph. An activity audit is appropriate for detecting when the Model approaches Structure Equilibrium. The Model's simulation can be left to run until it reaches Structure Equilibrium. This will not always be a global minimum: the final Structure Equilibrium might depend on

the simulation's starting state. An activity audit is appropriate in applications where all of the goals and constraints can be represented by external forces and adaptation rules. The 2D Fascia Model presented in Chapter 3 uses an activity audit.

2. Fitness Function - a measure to be optimized (ie minimized or maximized). Examples include the difference in Strain between a Model's Edges or the total mass of all the Edges and Nodes. Using the Attributes of the model to measure performance is often simpler and more appropriate than incorporating performance constraints. For example, it is easier to estimate cost mathematically from Attributes rather than attempting to incorporate rules and constraints to drive the cost down. The electrical grid Model presented in Section 4.3 is well suited for this method.

3. Performance Benchmarking - a simulation that evaluates the fitness of the model. Performance benchmarking is appropriate for complex goals like optimizing the structure's resilience to damage. In this case the evaluation is done as a separate simulation, subjecting the model to the conditions it is being optimized against and measuring its performance accordingly. For example, when optimizing a model for damage resilience, periodically pause the optimization and randomly remove components until the structure collapses; the more elements that can be removed before collapse, the better. The bridge Model presented in Section 4.2 is well suited for this method.

2.1.2 B - Graph Specification

Each tensegrity structure is defined as a Tensegrity Graph with Nodes and Edges. Nodes are defined as points in space. Edges connect pairs of Nodes and any pair of Nodes may be connected to at most one Edge. Each Edge has an attribute called Natural Length, which defines the optimal distance between its Nodes. Edges are labeled as Rods or Elastics, defining how they affect their connected Nodes. A Rod pushes its Nodes apart, to maintain the Rod's Natural Length. An Elastic pulls its Nodes together whenever the Elastic is stretched beyond its Natural Length.

Each Node has an Attribute called the Velocity Vector, which determines how the Node moves during simulation. A Velocity Vector is the sum of the Internal Forces enacted on the Node by the Rods and Elastics that connect to this Node, and any External Forces, like gravity. Nodes and Edges may have other Attributes depending on the needs of the Model. For example, if the Model is to simulate gravity, either the Nodes or the Edges must have an Attribute that represents Mass. External Forces, like gravity, are applied as Acceleration to the Nodes. The Acceleration may depend on Attributes (such as mass, in the case of gravity), or it may be constant. Any constraints to how many Rods and Elastics may share a Node, as well as the exact External Forces and Attributes, are Model specific.

2.1.3 C - Adaptation Rule Set

The framework's components are modular, allowing for freedom of experimentation. It is difficult to determine which Adaptation Rule Set is most appropriate for meeting the Optimization Goal of a particular Model. The Adaptation Rules that can be used depend on the Model's Graph Specification (Section 2.1.2). Their complexity and granularity depend on the Optimization Goal (Section 2.1.1). Instantiating a Model may involve setting the Optimization Goal and specifying the Tensegrity Graph and then iterating through many different Adaptation Rule Sets until something satisfactory is found. Similarly, the Optimization Goal and Adaptation Rules may be set while the Graph Specification is tinkered with. The following are guidelines to help users find a suitable Adaptation Rule Set to achieve the Model's Optimization Goal:

- Choosing Granularity of Attribute Values
  - Continuous - Use continuous attribute values whenever possible, because this allows for complex gradients of change and the best accuracy when it comes to

complex problems. The 2D Fascia Model presented in Chapter 3 uses Continuous Granularity.

  - Discrete - Use discrete attribute values when continuous granularity is unfeasible or provides an unnecessary level of complexity for the model. For example, when mimicking physical world problems, components are unable to fit the exact measurements, but commonly used sizes are freely available. The bridge Model presented in Section 4.2 is well suited for Discrete Granularity.
  - Binned - Use binned attribute values when component variability is a constraint. (ex: minimizing manufacturing costs for clothing by adhering to broad categories: Small, Medium, Large) The electrical grid Model presented in Section 4.3 is well suited for Binned Granularity.

- Choosing Type of Value Change
  - Multiplicative - Models using an Activity Audit should use multiplicative attribute changes (like adding 5 percent of the current value), because this avoids value stutter.
  - Additive - When using coarse attribute values (Binned and some Discrete), additive value changes are most appropriate and allow for additional fine tuning of the granularity by defining the size of a change step.

Whatever form the Adaptation Rules take, they should strive towards Structure Equilibrium, which means that rules should either come in increase/decrease pairs or they should allow for attribute scaling (Section 3.3). This allows for maximum automation.

2.2 Instantiating Models on the Framework

To instantiate a Model, the user must define its Components: the Optimization Goal (A - Section 2.1.1), the Graph Specification (B - Section 2.1.2), and the Adaptation Rule Set (C - Section 2.1.3). To illustrate this process in detail on a specific example, Chapter 3 covers the instantiation of a 2D Fascia Network Model. The versatility of the framework is presented in Chapter 4, with general guidelines for instantiation as well as several additional examples.

Chapter 3

Instantiating a 2D Fascial Network Model

This chapter provides a detailed example of specifying and instantiating a Tensegrity Model (Sections 3.1-3.3) and explores the utility of the example for contemporary research (Section 3.5). This example inputs a Tensegrity Graph and External Forces, and outputs the same Tensegrity Graph optimized for the provided External Forces. A correct solution is defined as a Tensegrity Graph whose Elastic Stiffness values are at the optimal ratio to the Strain the Elastics are under (Section 3.3).

The instantiation of the model proceeds by defining its three Components. The Optimization Goal (Section 3.1) is to simulate the performance of fascia, as it adapts to reach a state of Structure Equilibrium. The Tensegrity Graph Specification (Section 3.2) defines the Tensegrity Graph as a Rod and Elastic structure in 2D where rods represents bones (under compression) and elastics represent fascia (under tension). The Adaptation Rule Set (Section 3.3) specifies that fascia adapts by becoming stiffer in response to high strain.

An implementation of the Model has been coded in Unity - a game development engine with a simple UI for working in 3D vector space (all Z values are set to 0 in the implementation for ease of display), capable of displaying and editing variables throughout the simulation; the visualization provided by this implementation is illustrated with selected screenshots. The example in Figure 3.1 shows a fascial network with 6 Nodes: the four white

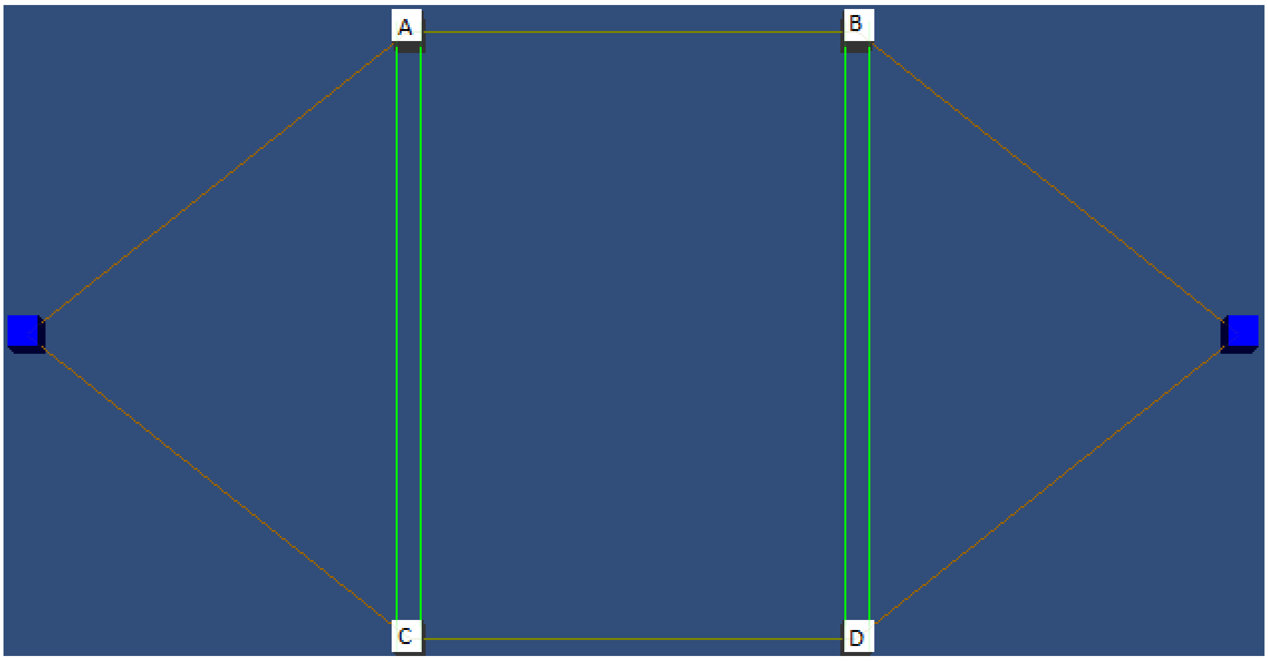

Figure 3.1: A screenshot from the Unity implementation of the 2D fascial network model. White Squares A,B,C & D are Nodes. Blue squares are nodes with fixed position. Single lines are Elastics. Multi-lines are Rods. The colour of the lines represents how much strain the Edges are under, with green being the least and red being the most. The Edge labels are stored in an adjacency matrix. The Stiffness values are stored in another adjacency matrix.

nodes (ABCD) are movable, while the two blue Nodes are fixed points in space. This example contains 8 edges: two rods drawn as multilines (connecting AC and BD), and six elastics drawn as single lines. The colour of the lines represents how much strain the Edges are under, with green being the least and red being the most. Thus, we see that in this configuration the rods are under very little strain, the elastics between AB and CD are under moderate strain, and the other four elastics are under the most strain. When the user indicates that the simulation should allow movement, tension on the elastics will be equalized by moving AC to the left and BD to the right. Adaptation will also occur; details of that are explained later.

Simulating a Tensegity Graph (Figure 3.1) requires a few extra steps to clear out the bias of the starting state. It is ideal to start a model in an Equilibrium state with an equal distribution of Strain throughout (ie everything adapted to the internal forces). To

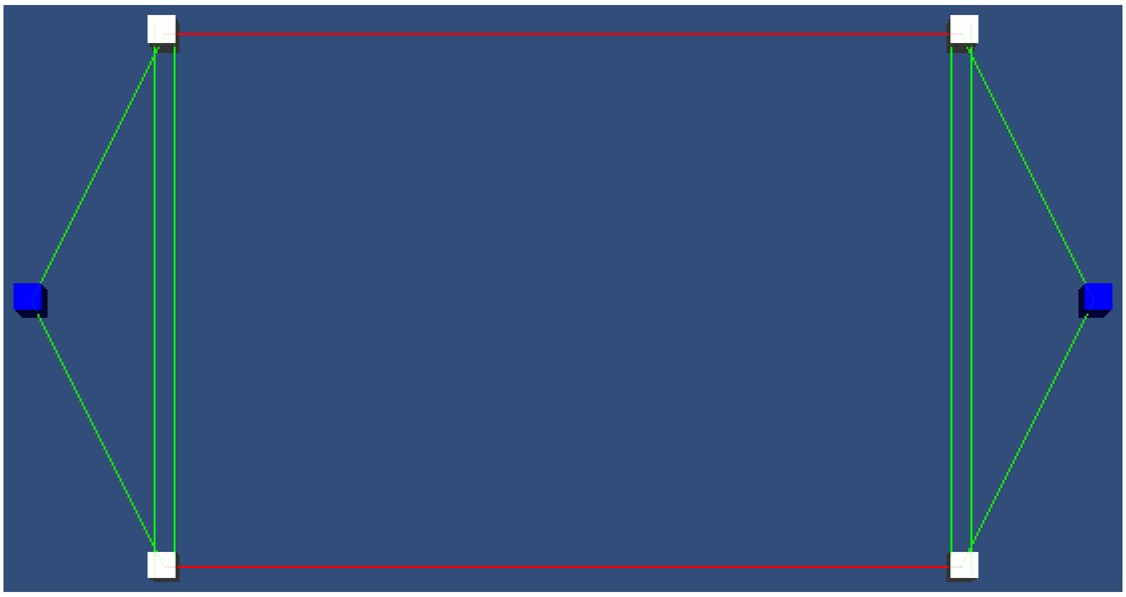

Figure 3.2: Tensegrity Graph at the start of simulation.

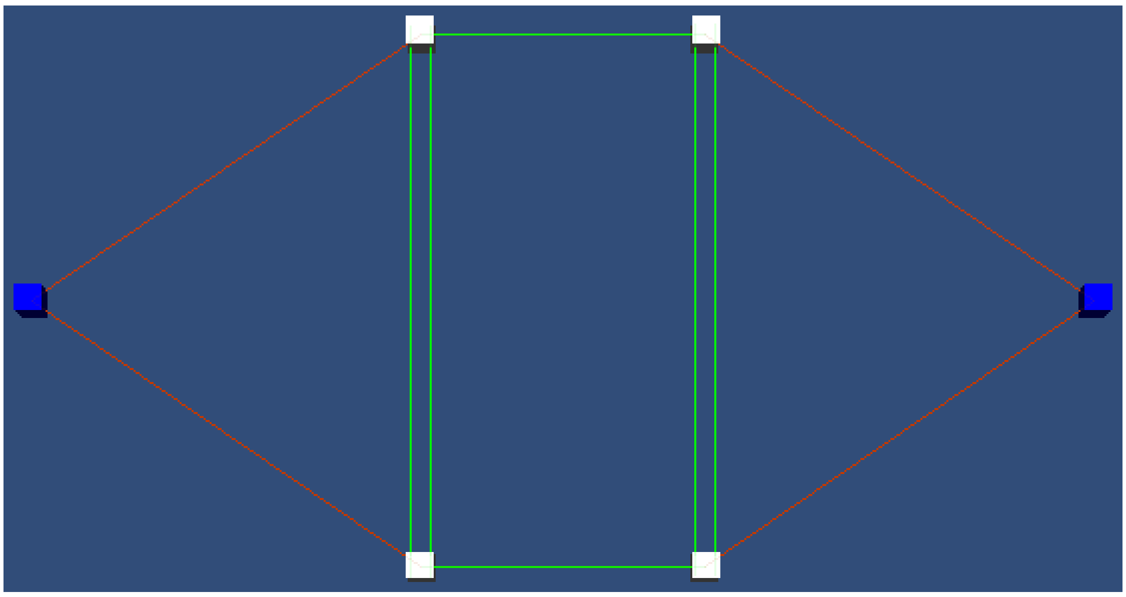

Figure 3.3: Tensegrity Graph in Structure Equilibrium based off of Internal Forces alone.

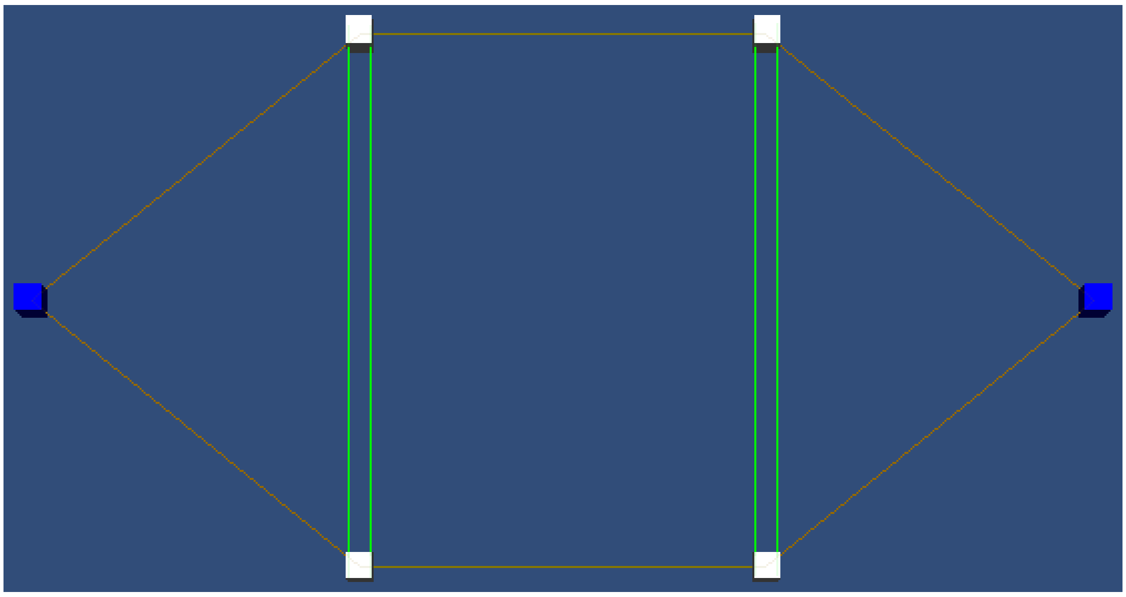
Figure 3.4: Tensegrity Graph in Structure Equilibrium after adapting to Internal Forces.

achieve this without hardcoding the values into the model (a time consuming matter) the initial instantiation happens without applying External Forces or the Adaptation Rule Set (Figure 3.2). This allows the Tensegrity Graph to settle into an Equilibrium state based on Internal Forces alone (Figure 3.3), clearing out any bias of the Nodes' starting positions. Then the Adaptation Rule Set is applied to the Graph, allowing the Edges to adapt their Attribute values to the Internal Forces as well, distributing the Strain as equally as possible throughout the Graph, and reaching another Equilibrium State (Figure 3.4). Then the External Forces are applied and the Graph moves towards the final Equilibrium State (Figures 3.5, 3.6, and 3.7). Applying the External Forces simultaneously with the Adaptation Rule Set can bias the adaptation towards adapting more quickly to the External Forces. This can affect the resulting equilibrium state.

Implementing the Model in Unity allows the user to take advantage of its UI, pausing the simulation at any time and changing Attribute values with real-time feedback. Figures 3.8 and 3.9 show the Unity UI being used to alter one of the Node Positions of the Tensegrity

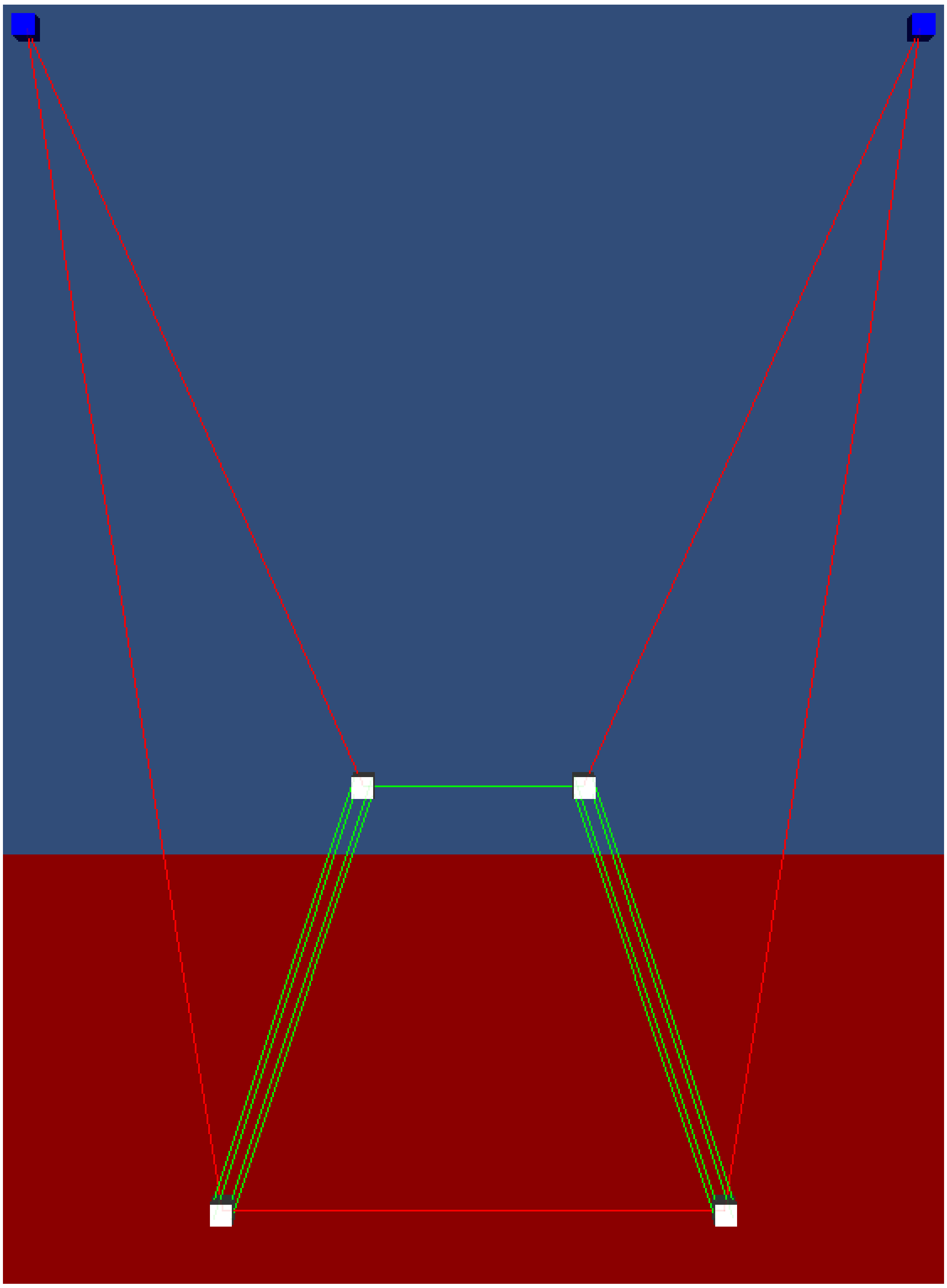

Figure 3.5: External Forces (simulating gravity by applying constant downward acceleration to the four white nodes) are introduced to a Tensegrity Graph adapted to Internal Forces. The Nodes start in the position seen in Figure 3.4 and move down.

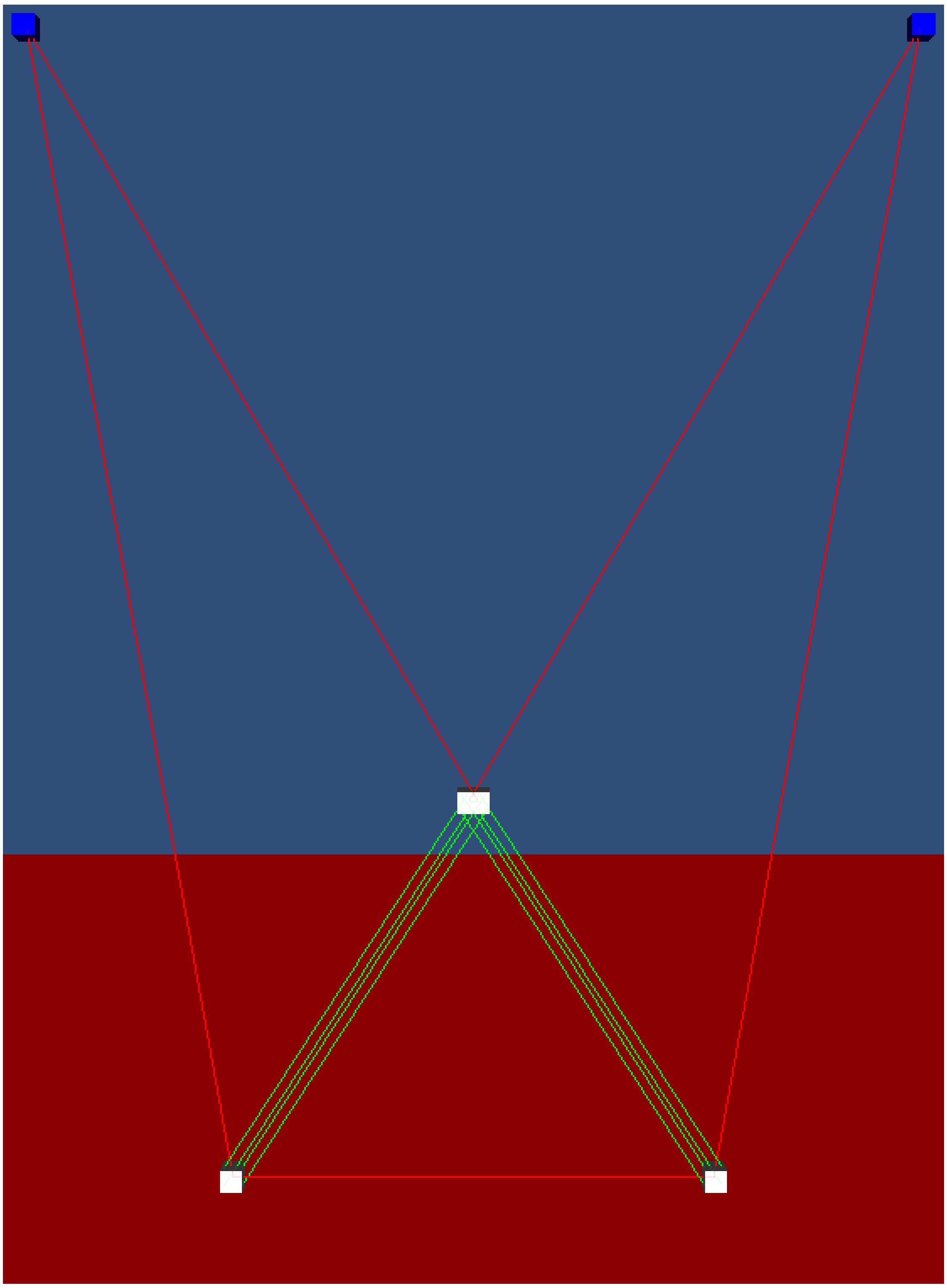

Figure 3.6: Tensegrity Graph midway through adapting to External Forces. The top white Nodes move towards each other as the rods pivot around the bottom white Nodes.

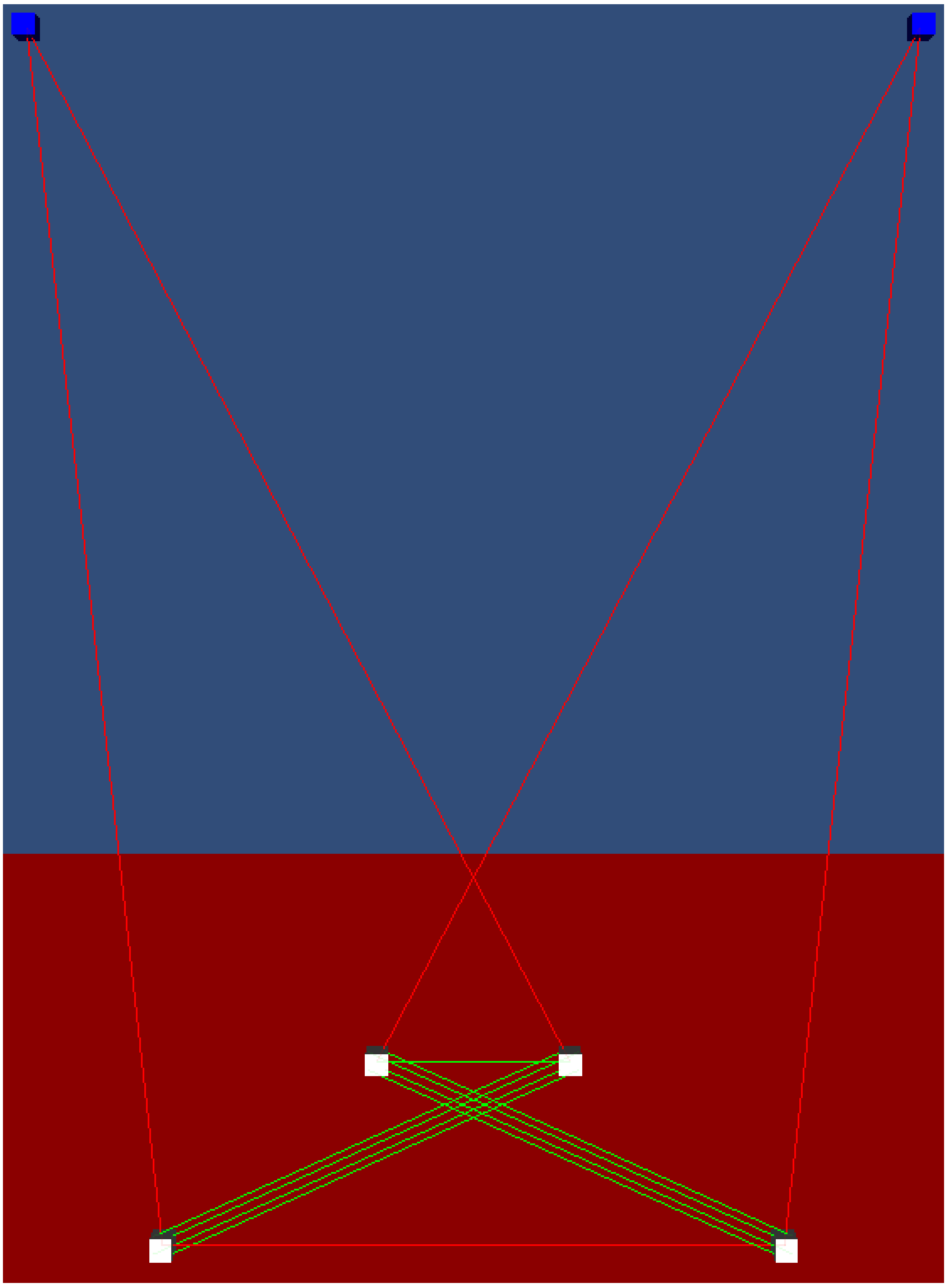

Figure 3.7: Tensegrity Graph in a State of Structure Equilibrium after adapting to Internal and External Forces.

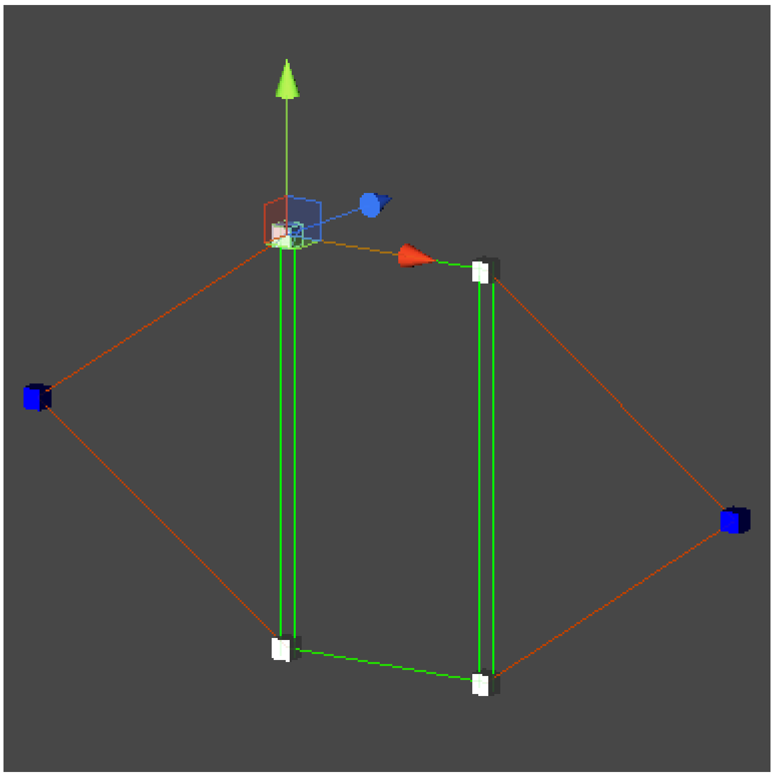

Figure 3.8: A Node of the Tensegrity Graph being selected in the Unity UI.

Graph illustrated in Figure 3.3, resulting in Figure 3.10.

3.1 A - Optimization Goal for this model of Myofascia

Fascia, also called *connective tissue*, supports the body, allowing body positions to be maintained without excessive muscular effort [15]. The tissue performs this by rapid adaptation to body configurations. The Optimization Goal of this Model is to simulate the performance of fascia as it reaches a State of Structure Equilibrium.

The fascia Model's Optimization Goal uses an Activity Audit (Section 2.1.1) to determine when the Tensegrity Graph has reached a State of Structure Equilibrium. This implementation sets a flag, signaling when the simulation may be ended or altered. The implementation also uses this flag to automate the introduction of gravity as an External

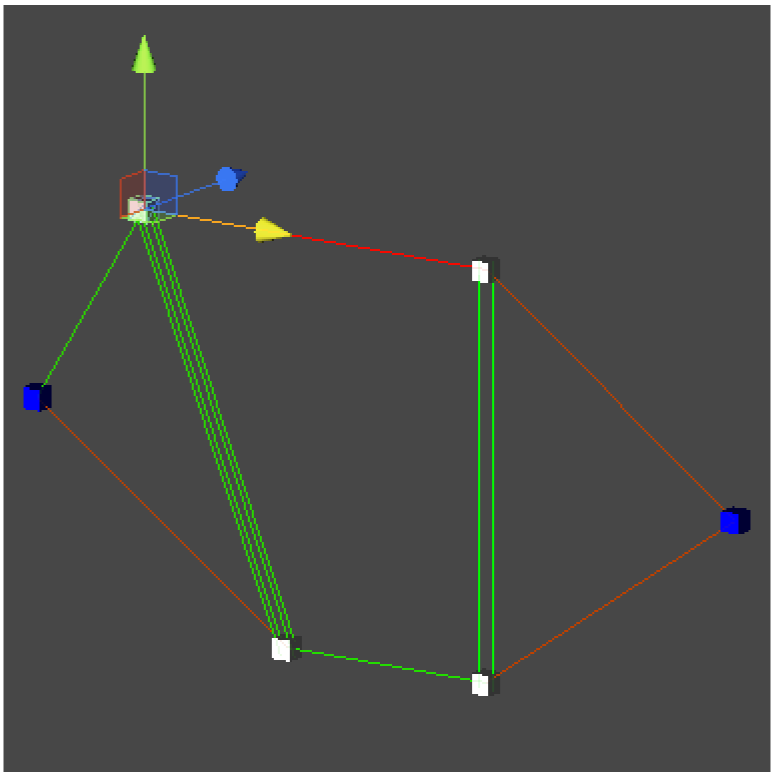
Figure 3.9: The selected Node Position has been altered from the position in Figure 3.3 to the position in Figure 3.10.

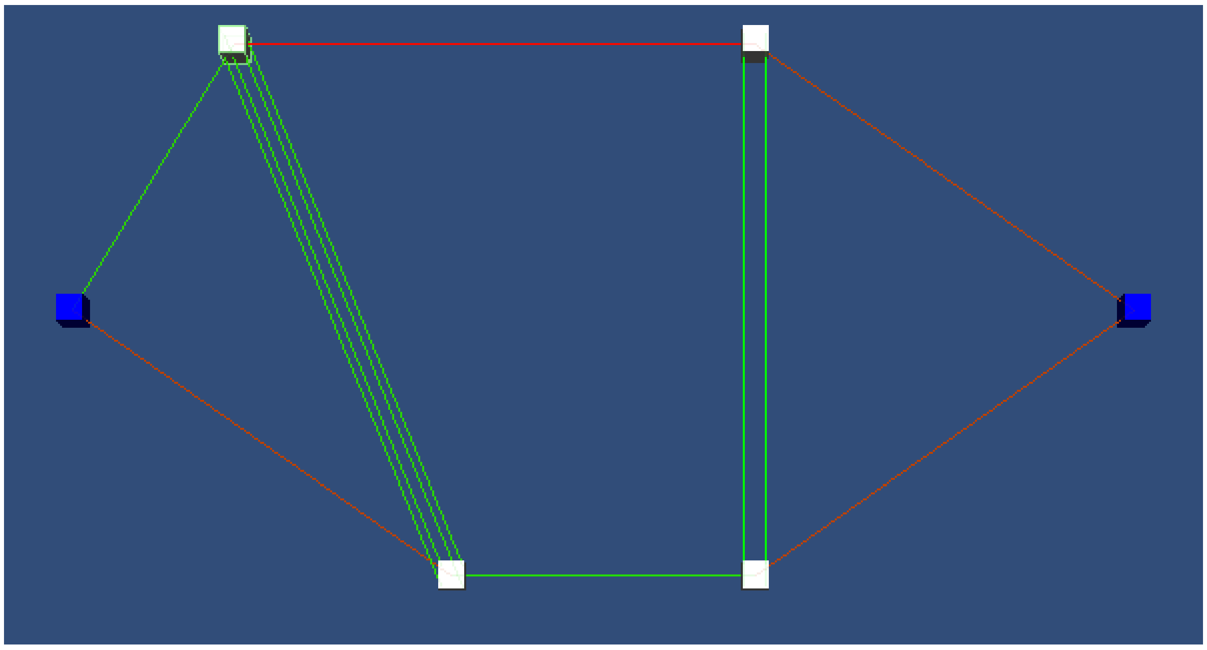
Figure 3.10: A Tensegrity Graph altered through the Unity UI.

Force, as well as to conduct a state space search to study how the starting Node Positions affect the resulting Structure Equilibrium States.

The Optimization Goal can be defined using one of the three methods in Section 2.1.1: Activity Audit, Fitness Function, or Performance Benchmarking. In this model, we choose to use an Activity Audit because it is the simplest of the 3 methods and signals when the Tensegrity Graph has reached an Equilibrium State. As an alternative, Performance Benchmarking could be used by removing Edges or augmenting the Activity Audit with random insertions of External Forces, similar to Simulated Annealing [1]. The downside of using Activity Audit is that it doesn't incorporate any method of traversing the search space of possible States beyond reaching a local minimum. Conversely that feature is also useful because we know that when an Activity Audit indicates that the nodes in the Tensegrity Graph are no longer moving, it has reached an Equilibrium State, a local minimum in the state space: the result can be evaluated (by introducing External Forces to observe the structural reaction, or by feeding the State through a Fitness Function to compare the effects of different starting positions) or the state space can be altered (by removing Edges to observe the structural reaction or introducing additional adaptation rules).

3.2 B - Tensegrity Graph Specification

A Tensegrity Graph is specified as a graph of Nodes, Edges, Edge-labels and Attributes, each representing an aspect of the structural optimization problem. The combination of generic and Model-specific specifications results in a connected bipartite Tensegrity Graph with no parallel Edges. Fascia is a tensional network [16] and maps neatly onto Elastic-labeled Edges. However, fascia in the body doesn't have compressed elements to give it structure because it is wrapped around organs and bones; thus the Rod-labeled Edges are an abstract representation of bone under compression.

With these abstractions, the fascia Model uses a Tensegrity Graph of the type illustrated in (Ex: Fig 1.1). Rods represent bones under compression (from the fascia wrapped around them), which push their Nodes apart to maintain their Natural Lengths. Elastics represent fascia under tension (from being wrapped around the bones); these Edges pull their Nodes together whenever they are stretched beyond their Natural Lengths. These abstractions are represented in the Tensegrity Graph specifications below.

Generic Specifications - The following is a list of generic specifications that hold true for all Models instantiated on the Framework:

- Any pair of Nodes may be connected by at most one Edge. The reasons for this restriction are as follows. Parallel Edges of the same label (Rod-Rod or Elastic-Elastic) would distribute the Internal Force generated on those Nodes between the two Edges and would adapt individually, but at identical rates. Parallel Edges of different labels (Rod-Elastic) would generate conflicting Internal Forces on their Nodes, which would cause the Edges to adapt to compensate for each other's interference. Depending on the Adaptation Rule Set, this may render one of the Edges unable to affect the Tensegrity Graph beyond hindering the other Edge, or it may cause a feedback loop, leading both Edges to adapt infinitely, making the instantiation meaningless.

- Edges have the Attribute *Natural Length* (length at which the Edge applies zero Internal Force to its Nodes). For this Model, the Natural Length of a Rod represents the length of a bone; deviations from this length cause very large Strain values, generating high Internal Force to correct the deviation. Similarly, the Natural Length of an Elastic represents the length at which fascia has no strain, and deviations from this length cause Strain proportional to the deviation, generating increasing Internal Force to correct the deviation.

- Nodes have the Attributes $Node\ Position$ (the Node's position in N-Dimensional space) and $Velocity\ Vector$ (the velocity of the Node, measured as a sum of Internal and External Forces applied to it). The fascia Model uses 2D space.

Model Specific Specifications - The following is a list of Model specific specifications that hold true for the fascia Model:

- Nodes may be rendered immobile to simulate a fixed position in space.
- Each Edge must either be Elastic-labeled (fascia under tension) or Rod-labeled (bone under compression).
- Elastic-labeled Edges have a *Stiffness* Attribute (the Internal Force generated by Elastics is directly proportional to their Stiffness). Rod-labeled Edges do not have a Stiffness Attribute (adaptation in bones [3] happens on a much slower scale than in fascia [13, 14]).
- Any number of Elastics may share a Node. For example, in Figure 3.1, Node A is connected to Node C with a Rod, Node B with an Elastic and a fixed node with an Elastic.
- Each Node must be connected to exactly one Rod.
- This Model uses two optional Edge Attributes: Current Strain (how much Strain the Edge is under at this time) and Integrity (how much Strain the Edge can withstand before breaking). In Figure 3.1 the Current Strain is visible by the colour of the Edges in relation to their Natural Length (green) and Integrity (red).
- Each Node is treated as having a mass = 1 for the purposes of F=mA in calculating Internal and External Forces (ex: gravity).

The current implementation of the Tensegrity Graph was intentionally kept small and simple, with the goal of supporting experimentation with the Adaptation Rule Set. The figures in this chapter are screenshots from the Unity model implemented for this thesis and some details of the implementation are described here. Figure 3.1 shows a starting position of a Tensegrity Graph: comprised of 6 Nodes (two with fixed positions), 2 Rods and 6 Elastics. The implementation stores Node Positions as 2D vectors in an indexed list. The values in the list are updated every simulation cycle based on the Velocity Vectors calculated during that simulation cycle. The two Nodes with fixed positions are included in the list, but their positions are not updated because they are skipped by the loop responsible for updating the Node Positions. The Edges and their labels are stored in an adjacency matrix indexed by Node id (corresponding with the Node's index in the Node Position list), while Stiffness and Integrity are recorded in another identically indexed adjacency matrix. The Edges' Natural Lengths are set to a static value of 1, while the Velocity Vectors are calculated at each simulation step from the Node Positions (to determine the direction of the Vector) and relative Strain values (to determine the magnitude of the vector).

3.3 C - Adaptation Rule Set

The ideal Stiffness : Strain ratio is defined by the user. In this example, the ideal ratio is 1:1 or Strain = Stiffness. The following rules are applied to the Edges with each cycle of simulation:

- If an Edge's Current Strain exceeds its Integrity (representative of being exposed to force sufficient to break it as defined by the model), the Integrity of this Edge is set to equal the current Strain.
- If an Elastic is more tense than ideal (Strain > Stiffness), its Stiffness is increased by 5 percent of the Stiffness increase which would be necessary to totally compensate for the deviation from ideal. [Stiffness[t+1] += (Strain[t] - Stiffness [t]) * 0.05]
- If an Elastic is less tense than ideal (Strain < Stiffness), its Stiffness is decreased by

5 percent of the Stiffness decrease which would be necessary to totally compensate for the deviation from ideal. [Stiffness[t+1] -= (Stiffness [t] - Strain[t]) * 0.05]

As the Stiffness of tense Elastics is increased with each simulation step, the Internal Forces they apply on their Nodes increases, bringing them closer. For example, in Fig 3.1, Rods AC and BD will move away from each other because of the inequality in Strain in their elastics, but as the Elastics adapt and their Stiffness values change, the Strain values will balance out as well. This forms a negative feedback loop, affecting the rate of change of an Elastic's Stiffness by affecting its Current Strain. As mentioned in the beginning of this chapter, the starting positions of the Nodes biases the adaptation if the Adaptation Rule Set is applied at the same time as the simulation starts. To clear out this bias, instantiations of the 2D Fascial Network Model begin without applying External Forces or the Adaptation Rule Set. The Tensegrity Graph is allowed to find Structure Equilibrium with Internal Forces alone. This doesn't guarantee finding the global minimum (explained in Figure 3.3), but it finds the closest local minimum (which may be the global minimum). Once, the Activity Audit signals that Structure Equilibrium was found, the Adaptation Rule Set is applied and the Tensegrity Graph is allowed to adapt to the Internal Forces. At this point, most of the bias stemming from the starting Node Positions is cleared out and the simulation can introduce External Forces. While the Adaptation Rule Set is applied, the structure compensates for any Internal and External Forces it is subjected to. The Integrity Attribute confers insight to how much tougher the Edges would have to be in order to withstand the Forces they were exposed to through all the steps. Without the gradual introduction of Adaptation Rules and External Forces, the differences in Stiffness would have been reflected in much higher increases of Integrity.

The Integrity Adaptation Rule is essentially a max function - if the new value is higher, update the value. The pair of Elastic Adaptation Rules were implemented by a single function: Stiffness[t+1] += (Strain[t] - Stiffness [t]) * 0.05. 5 percent is an arbitrary value, set low for the sake of slow, observable adaptation. This is roughly analogous to the learning rate in neural networks: the value can be changed or even included as part of the Graph Specification to study its effects on the Model. The learning speed also imposes a degree of realism, because fascia also possesses a limited adaptation rate [13, 14]. Being exposed to varying forces leaves fascia (actual and modeled) adapting towards the most current configuration of strain. If that adaptation opposes prior adaptation, that adaptation is undone. However, forces encountered by fascia in real life are rarely exact opposites, so the resulting adaptation is closer to an aggregate of encountered forces, weighted by chronology, intensity and duration [2, 9].

3.4 Bias Caused by Starting Node Positions

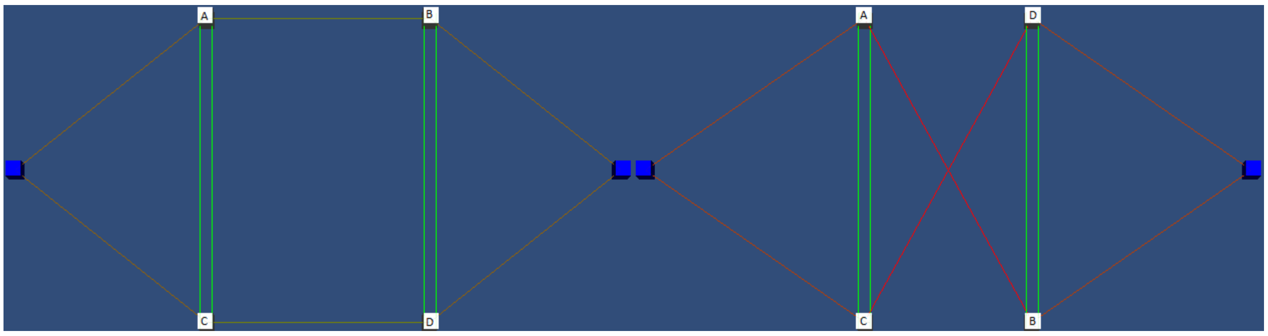

Figure 3.11: Both graphs have identical connectivity, but differ in starting Node Positions. As a result, these two graphs settle into differing States of Structure Equilibrium. The Graph on the left has a lower total Strain among the Edges. Because the Model is instantiated in 2D space, the Graph on the right is unable to settle into the same State as the graph on the left.

The fascia Model described here is not guaranteed to find a global minimum. The local or global minimum that is found depends on the starting Node Positions. This is illustrated by Figure 3.11. In both configurations, the Strain and Internal Forces generated by Elastic AB are identical to those of Elastic CD. This means that, despite it being possible to simulate the rotation of Rod BD, the Internal Forces generated within the Tensegrity Graph are not sufficient to escape the local minimum. To do that, user interference or External Forces would be required.

3.5 Emergent Properties of Fascial and Neural Networks

The above Model presents a novel way of computing emergent feedback loops. A similar simulation could have been encoded into a neural network [10], but not without significant additional complexity. Each Node would need a neuron in the input layer with connections to stimulations representing the aggregate Internal Forces and simulated External Forces. Each Edge would require a neuron on the inner layer and a variable weight connection to represent its Attributes and Internal Forces. In contrast, most of these important aspects of the simulation are integral to our Structure Optimization Framework: the Model's Components provide a straightforward way to instantiate and simulate a complex distributed structure.

The most important difference between a Tensegrity based Fascial Network and a Neural Network is the nature of the graphs that describe them. A Neural Network is represented as a directed graph [5], while a Tensegrity Graph is by necessity undirected. This fundamental difference is the biggest source of incomparability between the functions of the two types of networks. Both support supervised and unsupervised learning, continuous and discrete learning, directed and undirected activation, but to different degrees. Since each type of network is specialized for its own niche, it is easier to study Models of their corresponding niche rather than translating the Models to the other network type. It would be possible to instantiate an unsupervised, continuous learning model on an undirected neural network, to simulate biological adaptation/learning. Such a model would share many aspects with the 2D Fascial Network Model and may very well simulate the adaptation that takes place in the biological system to an equivalent or superior degree. The drawback would be an immense increase in complexity and translation because neural networks are best suited for supervised, discrete learning and are most often instantiated as a directed activation network with an input layer, a hidden layer and an output layer.

The fascia Model presented in this chapter possesses two emergent properties that also occur in Neural Networks: distributed memory and graceful degradation. Compared with Neural Networks, the distributed memory in the fascia Model is fairly rudimentary - the State of the Tensegrity Graph is stored in the Attribute values of the Edges and Nodes. However, the graceful degradation exhibited by the system is on par (if not superior to) that of Neural Networks. The Tensegrity Graph can recover lost Attribute values through the Adaptation Rule Set, compensating for the sudden imbalance of Internal Forces that results. The removal of Edges can be compensated for and optimized against by the same mechanism.In contrast, Neural Networks (simulated and biological) have critical nodes, the removal of which can cripple the performance of the system; examples are the removal of an important node in the hidden layer of a neural network or a brain lesion to a critical structure.

Neural Networks were designed to mimic the adaptation that happens in the brain and peripheral nervous system. The Fascia Model was designed to mimic the adaptation that happens in the bones and connective tissues. Essentially, Neural Networks simulate brain learning, while the fascia model simulates body learning. Studying biological adaptation should include both. This topic is discussed further in Section 5.9.

Chapter 4

Versatility of the Framework

This chapter provides guidelines for instantiating and optimizing models (Section 4.1). We present two additional example models (bridge structure in Section 4.2 and electricity grid in Section 4.3) to support the detailed fascia Model in Chapter 3. The chapter concludes by exploring the characteristics of optimization problems that are suitable for modeling on the framework (Section 4.4).

4.1 Guidelines for Instantiating and Simulating Models

- The Optimization Goal (A - Section 2.1.1) is defined as a desired end State for the Tensegrity Graph by one of the methods detailed in Section 2.1.1 and simulation terminates when that State is reached. The most suitable method for defining the Optimization Goal depends on the type of model. The fascia Model (Chapter 3) uses an Activity Audit; a fitness function would be less suitable for this model because the optimization of the whole structure is difficult to express as a function of its parts. The bridge structure Model uses performance benchmarking (Section 4.2.1), and the electricity grid model uses a fitness function (Section 4.3.1).
- The Graph Specification (B - Section 2.1.2) defines the Edge labels, Edge Attributes, and Node Attributes used in a Tensegrity Graph. Models use different Attributes and labels to represent different things, depending on their modeling needs. For example, the fascia Model uses the Strain Attribute to represent physical strain on fascia or

bone, whereas the electricity grid Model (Section 4.3) uses Strain to represent the flow of electricity through a cable. The Graph Specification may also include constraints on the allowable connectivity in a Tensegrity Graph.

- The Adaptation Rule Set (C - Section 2.1.3) is implemented as Attribute changes that are designed to move the Tensegrity Graph's State closer to the Optimization Goal (A). The Attribute changes do not have to further the optimization. For example, the Bridge Structure (Section 4.2) uses its Adaptation Rule Set to simulate the natural deterioration of the bridge.

4.2 Bridge Structure

A bridge presents an obvious structural optimization problem: the bridge must bear its own weight and the weight of anything crossing it, rain, sleet or snow. In this example, the user is evaluating a number of bridge designs and needs to figure out which design would last longer under various simulated conditions. If an Edge breaks then the bridge is considered to have reached the end of its lifespan and the simulation ends. The Model incorporates various performance conditions (ex: gravity, vehicle traffic, weather conditions) as External Forces and represents the effects of time (warping under continuous strain, degradation due to weathering, aging and rusting) as Adaptation Rules.

4.2.1 A - Optimization Goal for this model of Bridge Structures

The Optimization Goal for this Model is to maximize the structure's longevity. Section 2.1.1 details several methods of measuring a Model's performance: an activity audit, a fitness function or performance benchmarking. An activity audit is not suited to this Model because bridges are largely immobile, so the threshold for the audit would have to be very low. Performance benchmarking is the most fitting evaluation method because the bridge

Model has numerous testing parameters. Alternatively, a fitness function could be used; that would allow the user to incorporate values not represented by Attributes, such as cost or estimated construction time.

4.2.2 B - Tensegrity Graph Specification

The bridge Model uses 3D space: each Node has X, Y, Z Attributes representing the position of the node. In this model, all edges have the label *Rod*. Nodes represent the connection points between the bridge's parts, while the components themselves are represented by Rod-labeled Edges. Each Edge has the optional Attributes *Integrity*, *Strain*, and *Density*. The mass for each Edge is calculated as the Edge's Natural Length multiplied by the density of the material the Edge represents. If the Strain of any Edge exceeds its Integrity, that Edge will need to be replaced or repaired, marking the end of the structure's longevity for this model.

4.2.3 C - Adaptation Rule Set

The bridge model uses the Adaptation Rule Set to simulate the deterioration of the structure. At each simulation step, the Integrity of each Edge decreases by an amount proportional to its current Strain. Save for the deterioration, the bridge is effectively a static structure and any design flaws will be magnified over time by the increasing imbalance of Strain among the Edges. The simulation ends when the Strain of some Edge exceeds its Integrity, signaling that this Edge has reached its breaking point. The effect of losing an Edge will have immediate repercussions on the rest of the structure, as a delicate balance of Internal and External forces sharply shifts. The Edge that breaks has the potential to cause a chain reaction, bringing down the whole bridge.

4.3 Electricity Grid

An electricity grid presents a more abstract structural optimization problem than the bridge problem in Section 4.2. This Model requires a rather abstract mapping to capture the relevant aspects of the flow of electricity as a Tensegrity Graph. The electricity grid Model uses 3D space: each Node has X, Y, Z Attributes, where (X,Y) represents the spatial location and Z represents the draw or production of electricity at this Node. Movement along the X and Y axes is impossible as the infrastructure of an electricity grid is immobile. The coordinates are used to calculate Internal Forces, but as a multiplier on the differences in Z values. In this model, Rod-labeled Edges represent electricity draw and consumption, and Elastic-labeled Edges represent cabling; this is described in greater detail in Section 4.3.2. The movement of electricity is represented by Strain, as higher levels of Strain and geographical distance will disperse farther, representing the limitations of moving electricity large distances. Such a Model is useful as a simulation tool to predict consequences of damage, and to locate critical points within a larger infrastructure.

4.3.1 A - Optimization Goal for this model of Electrical Grids

In the electrical grid Model, the Optimization Goal is to minimize cost without allowing failure. Section 2.1.1 details several methods of measuring a Model's performance: an activity audit, a fitness function or performance benchmarking. An activity audit is not suited to this Model because the electrical infrastructure is largely immobile, so the threshold for the audit would have to be very low. Large scale civic projects are very expensive and time consuming, so a fitness function is used to take those factors into account. To ensure that the structure does not fail, the user can simulate peak draw levels as External Forces. Optional Edge Attributes are used to simulate electrical draw on an individual Edge (Strain Attribute) as well as the upper limit (Integrity Attribute). Each potential solution State generated by the Model's instantiation is fed through a fitness function to determine the cost of such a structure. To qualify as a potential solution State, the electrical draw of every location must be met and no location may exceed its maximum electrical production. The fitness function can be as simple as totaling the cost of all components or as complicated as calculating average maintenance costs for each line. The simulation terminates when the potential solution States fed through the fitness function start repeating or when the user terminates the simulation, whichever comes first.

4.3.2 B - Tensegrity Graph Specification

The electrical grid Model uses a 3D space with the X and Y axes representing locations in space and the Z axis representing the draw (-) and production (+) of electricity. The Graph Specification uses Rod and Elastic as Edge labels, as well as the Stiffness optional Attribute. The Model's Attributes, Edges and Nodes are representative abstractions of the variables at play in a physical electrical grid:

- Nodes represent infrastructure that either produces, distributes or consumes electricity.
- Node Positions represent the physical location of the node. The X/Y values are fixed to represent geographical coordinates. The range of Z values depends on the type of infrastructure. Producing Nodes cannot have negative Z values. Consuming Nodes cannot have positive Z values. Distributing Nodes can have positive or negative Z values, but may not be connected to a Rod.
- Rods connect Nodes to the X/Y plane, representing how much electricity they draw (consuming Nodes) or produce (producing Nodes).
- A Rod's Natural Length represents the maximum production/peak draw of the infrastructure the Rod represents.

- Elastics connect Nodes together, representing the cabling between infrastructure. Producing Nodes may not be connected directly to consuming Nodes. Distributing Nodes may connect to any type of Node.

- Positive acceleration along the Z axis represents electrical production and the satiation of draw, while negative acceleration represents electrical draw at locations. Acceleration only occurs on the Z axis (because infrastructure does not move), but is affected by the physical distance (Euclidean distance calculated from X/Y values) and Elastic Stiffness.

- An Elastic's Stiffness modifies the Internal Forces it generates, increasing any positive Z acceleration and decreasing any negative Z acceleration to represent more efficient satiation of electrical draw.

The Internal Forces generated by Edges represent the imbalance of electricity. Rods push producing Nodes above the X/Y plane and push consuming Nodes below the X/Y plane. The net effect of the spanning tree of Elastics that connects all the Nodes is that the Nodes are all pulled towards the X/Y plane.

4.3.3 C - Adaptation Rule Set

Cabling has to be manufactured in bulk, reducing the available choices down to a few categories of cable. With only a few types of cable available to the project, the user declares and sorts the available varieties by price: Low, Medium and High. The price of the cable is proportional to its efficiency, represented by the Stiffness. The user then defines a simple Adaptation Rule Set:

- If a Rod's Strain exceeds its Integrity, increase the quality of all the Elastics connected to its node by one category.

- If the previous rule would increase an Elastic's quality above High, select a random Elastic connected to the same distributing Node and increase its quality by one category.
- If all consuming Node Positions have Z= 0, reduce the quality of some Elastic by one category.

The first rule ensures that the Elastics connected to the producing Nodes are of sufficiently high quality. The second rule creates a sort of spreading activation, ensuring that the shortest lines between the producing and heavy consuming Nodes are connected by the best required quality Elastics. The last rule drives down the cost, since the first 2 rules only establish a functional baseline configuration. The user can perform the 3rd rule manually to reduce the overall cost. Whether the rule is implemented automatically or manually, any time a reduction in quality results in an invalid State, the first two rules ensure that the grid is returned back to a valid State.

4.4 Determining if an Optimization Problem is Suitable for the Framework

The Optimization Framework presented in this thesis is based on physical Tensegrity structures. While it is possible to translate abstract systems into a representative Tensegrity Graph (as Section 4.3 demonstrates), the effort of translation can be a significant cost in constructing the Model. Thus, the optimization problems most suitable for instantiating on the Framework are physical systems that have too many connected, tightly coupled parts to allow easy simulation with other methods. The decision of whether an optimization problem is worthwhile to Model on the Framework becomes an optimization problem of its own, balancing the benefits of unsupervised optimization versus translation time and efficiency.

Chapter 5

Conclusion

This chapter summarizes the contributions of this thesis, as well as avenues for future work. The major contributions of this thesis are a structure optimization framework (Chapter 2), a detailed example of instantiating the framework to model fascia optimization (Chapter 3), additional examples of framework instantiation (Chapter 4), and the use of the fascia optimization model as a basis for studying emergent properties (Section 3.5). Section 1.5 listed the following desirable properties for these contributions:

- generic
- applicable
- intuitive
- useful
- interactive
- rapid prototyping
- insight
- fidelity
- alternative to neural networks for studying emergent properties (structural memory, graceful degradation, continuous learning, undirected activation)

Below we discuss each property in turn, summarizing the evidence that the thesis work achieves this property (at least to some degree), and mentioning future research directions.

5.1 Generic

The Structure Optimization Framework is ***generic***, meaning that it can be used to model a wide variety of physical systems. The combination of graph theory, Attributes and Edge labels allows for mapping a wide variety of structural optimization problems onto the Tensegrity Framework, as evidenced by the fascia Model of Chapter 3, the bridge Model of Section 4.2 and the electricity grid Model of Section 4.3.

Each Model benefits from a custom set of Attributes and labels to best describe the functionality of the system it abstracts. Refining and instantiating the Models presented here is a great avenue for future work, particularly for the bridge and electricity grid Models which were presented at a conceptual level. Another avenue of future work is to investigate whether it is possible to model a restricted form of a classical NP-complete problem on the framework.

5.2 Applicable

The Structure Optimization Framework is ***applicable***, meaning that it can be used to create effective abstractions a wide variety of physical systems, representing only those aspects of the system that are required for optimization. This is evidenced by the examples in Chapter 3 and Sections 4.2 and 4.3. These three examples are very different in scope and application, but are instantiated easily as Models of the Structure Optimization Framework.

An important avenue for future work is to formulate formal validation procedures.

Validation becomes more difficult with more abstract models, but a low abstraction model, like the fascia model, can be validated against existing physical models.

5.3 Intuitive

The Structure Optimization Framework is ***intuitive***, as evidenced by Chapter 4 which provides a clear and straightforward method of instantiating optimization problems as models on the framework. Rendering the simulation visually in real time; as was done for the fascia Model in Chapter 3, helps the user to understand what is happening to the model as it reaches Structure Equilibrium.

5.4 Useful

The Structure Optimization Framework is ***useful***, in that it is faster to define some systems and their adaptation rules in the framework than to use other methods. Section 3.5 details the advantages of simulating optimization problems within the Structure Optimization Framework as opposed to a neural network. Other optimization methods exist of course - finite element modeling, for example - but they may require a substantial investment in specifying details of the modeling. The Structure Optimization Framework supports structure-based simulations at a high level of abstraction.

5.5 Interactive

The Structure Optimization Framework is ***interactive***: it can be modified mid-simulation to examine resulting behavioral changes. The 2D Fascial Network Model of Chapter 3 was rendered in Unity, which provided a very simple UI to interact manually with the simulation, allowing for pausing, altering the Node Positions and examining the effects on Edge Strain those alterations have immediately and over time after the simulation is unpaused.

5.6 Rapid Prototyping

The Structure Optimization Framework allows for ***rapid prototyping***, since it is quick to alter the Graph Specification and Adaptation Rule Set of a Model. This was particularly apparent during the implementation of the 2D Fascial Network Model of Chapter 3: the Adaptations rules were single lines of code and the entire Tensegrity Graph was stored in 2 adjacency matrices and a vector list.

5.7 Insight

The Structure Optimization Framework provides novel ***insight***. Because of the high degree of abstraction employed, the computational requirements are minimal, allowing users to run multiple simulations in parallel to better illustrate the differences between parameters. This approach enables users to do in depth studies of optimization processes. For example, the fascia model of Chapter 3 can be instantiated multiple times with different paramenters (e.g. different Tensegrity Graphs or different adaptation rules), and the user can determine which instantiation best matches the characteristics of fascial plasticity observed in biological systems [13, 14].

5.8 Fidelity

The Structure Optimization Framework maintains ***fidelity*** to real world systems. Instantiated Models can accurately simulate the behaviors observed in real world systems. These can be elementary behaviors, such as the strain generated by objects, or complex emergent properties whose exact causes are outside of the scope of the simulation. Fidelity is limited by the scope of the Model and the detail provided by the user, but nothing in the design of the Structure Optimization Framework places ceilings on the potential fidelity of an instantiated Model.

Any given Model can be refined to give greater fidelity in simulating the system it abstracts. Refining and instantiating the Models presented here is a great avenue for future work. For example, the 2D Fascial Network Model of Chapter 3 could be refined by simulating 3D structures, adding the capacity to detect collisions between edges of the Tensegrity Structure, simulating changes in bone strength in response to strain [3], modeling muscles and nerves, and simulating more complex structural anatomy like a cranium [11] or helices [12]. Another avenue of future work is adding support for NASA's Tensegrity Robotics, which use actuators in elastics.

5.9 Alternative to neural networks for studying emergent properties

The Fascial Network Model from Chapter 3 provides an alternative to neural networks for studying emergent properties by supporting a novel memory system, graceful degradation, continuous learning and undirected activation - qualities that are difficult to achieve in a neural network.

Neural Networks were designed to mimic the adaptation that happens in the brain and peripheral nervous system [5]. The Fascia Model was designed to mimic the adaptation that happens in the bones and connective tissues. Essentially, Neural Networks simulate brain learning, while the fascia model simulates body learning.

Conventional neural networks can be represented as directed graphs and they typically perform supervised learning algorithms. Unsupervised learning algorithms for neural networks (such as the original algorithm devised by the father of neural networks - Hebbian learning [6]) are more difficult to use for practical problems, because their input data needs to be carefully sculpted to prevent overlearning and underlearning. In contrast to the supervised learing on directed graphs done by a neural network, the fascial network presented in Chapter 3 performs continuous unsupervised learning on an undirected graph. Both neural networks and the fascial network Model demonstrate a memory system which supports graceful degradation.

This contribution is the most important source of future work from this thesis. As mentioned in Section 3.5, Neural Networks and Fascial Networks both simulate biological adaptation, but from different approaches, each with its benefits and drawbacks. Biological adaptation should be studied with both of these network systems in concert since the systems they are best suited to model frequently interact in as yet poorly understood ways [8, 13, 14]. Novel forms of memory can also be abstracted in Fascial Networks, such as the 'encountered problems' system of memory utilized by immune systems [7].

5.9.1 Continuous Learning

The Fascial Network Model uses ***continuous learning***. The Adaptation Rule Set presented in Section 3.3 uses the current values of the changing variables to compute how much they will change, simulating the diminishing returns in learning speed demonstrated by many systems when presented with stimuli requiring small adaptation and stimuli requiring large adaptation.

A great area for future work is adding complexity to the learning rules to more accurately simulate the learning speeds of various situations, including biasing them towards particular goals. For example, muscles develop with heavy use more slowly than they atrophy with underuse due to the high energy cost of maintaining high muscle mass [18].

5.9.2 Undirected Activation

The Fascial Network Model uses ***undirected activation***. Every Node in the Graph Specification is a point of input for the simulation, as the Node Positions relative to other Nodes determines how much Internal Force their connected Edges generate. This is contrasted by the directed nature of Neural Networks, which necessitates that input be applied to a layer of input neurons.

5.9.3 Structural Memory

The Fascial Network Model has ***structural memory***: the State of the Tensegrity Graph is recorded entirely in the Attributes and connections. The Attribute values change with time as the structure optimizes to encountered input; thus the memory of the whole State is distributed throughout the graph.

By implementing an asymmetric Adaptation Rule Set in future work, it is possible to bias the development of the structure to represent different memory storage systems. The Adaptation Rule Set presented in Section 3.3 is a short term aggregate memory system, and will always optimize for the currently encountered forces at a set speed. This also means that if it is presented with 2 alternating configurations of external forces and allowed to optimize for each before switching to the other, the Model will adapt infinitely between two states, instead of optimizing for the average of both configurations. This could be solved by biasing the adaptation of the structure with historical data, essentially averaging current inputs with previous ones to find an overall average. This creates a spectrum of input memory to study in future work, between perfect recall of all encountered states and the currently implemented zero recall. Interesting configurations could be medium term buffers that only consider the last N states, or a weighting system to organize and contrast previously encountered states by magnitude of importance.

5.9.4 Graceful Degradation

The Fascial Network Model supports ***graceful degradation***. When Edges or Nodes are removed from the Tensegrity Graph, the model can continue to function and adapt. The distribution of strain and Internal Forces changes to reflect the new structure, and new configurations may result in further Edges being removed, but the structure does not fail outright from a single broken part. This feature can be used to manually identify critical components in the structure or to measure the performance loss from local injury.

Neural Networks were designed to mimic the adaptation that happens in the brain and peripheral nervous system. The Fascia Model was designed to mimic the adaptation that happens in the bones and connective tissues. Essentially, Neural Networks simulate brain learning, while the fascia model simulated body learning. They are both capable of many similar functions, and both support functions of which the other is incapable, but both simulate biological learning in their different way.

Bibliography

[1] Thomas H Cormen, Charses E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to Algorithms.* The MIT Press, Cambridge, 3rd edition, 2009.

[2] Simona-Mariana Cretu. Tensegrity as a structural framework in life sciences and bioengineering. In *Modeling, Simulation and Control of Nonlinear Engineering Dynamical Systems*, pages 301–311. Springer, 2009.

[3] Harold M Frost. Wolff's law and bone's structural adaptations to mechanical usage: An overview for clinicians. *The Angle Orthodontist*, 64(3):175–188, 1994.

[4] Richard Buckminster Fuller. *Synergetics.* Macmillan, New York, 1975.

[5] Simon Haykin. *Neural Networks: A comprehensive foundation.* Prentice Hall, 1994.

[6] Donald Olding Hebb. *The Organization of Behavior: A Neuropsychological Theory.* Psychology Press, 2002.

[7] Charles A Janeway, Paul Travers, Mark Walport, and Mark J Shlomchik. *Immunobiology.* Garland Science, New York, 6th edition, 2005.

[8] John W Krakauer. Motor learning: Its relevance to stroke recovery and neurorehabilitation. *Current Opinion in Neurology*, 19(1):84–90, 2006.

[9] Stephen M Levin. The importance of soft tissues for structural support of the body, 1995. Retrieved from http://www.biotensegrity.com/

[10] Arianna Maffei and Gina G Turrigiano. Multiple modes of network homeostasis in visual cortical layer 2/3. *The Journal of Neuroscience*, 28(17):4377–4384, 2008.

[11] Graham Scarr. A model of the cranial vault as a tensegrity structure, and its significance to normal and abnormal cranial development. *International Journal of Osteopathic Medicine*, 11(3):80–89, 2008.

[12] Graham Scarr. Helical tensegrity as a structural mechanism in human anatomy. *International Journal of Osteopathic Medicine*, 14(1):24–32, 2011.

[13] Robert Schleip. Fascial plasticity – a new neurobiological explanation: Part 1. *Journal of Bodywork Movement Therapies*, 7(1):11–19, 2003.

[14] Robert Schleip. Fascial plasticity – a new neurobiological explanation: Part 2. *Journal of Bodywork Movement Therapies*, 7(2):104–116, 2003.

[15] Robert Schleip. Introduction. In *Fascia: The Tensional Network of the Human Body*, pages xv–xviii. Churchill Livingstone, Edinburgh, 2012.

[16] Robert Schleip, Thomas W. Findley, Leon Chaitow, and Peter A Huijing, editors. *Fascia: The Tensional Network of the Human Body*. Churchill Livingstone, Edinburgh, 2012.

[17] Robert Schleip, Heike Jager, and Werner Klingler. Fascia is alive: How cells modulate the tonicity and architecture of fascial tissues. In *Fascia: The Tensional Network of the Human Body*, pages 157–163. Churchill Livingstone, Edinburgh, 2012.

[18] Stephan Sorichter, Johannes Mair, Arnold Koller, Peter Secnik, Wojtech Parrack, Christian Haidand, Erich Muller, and Bernd Puschendorf. Muscular adaptation and strength during the early phase of eccentric training: Influence of the training frequency. *Medicine and Science in Sports Exercise*, 29(12):1646–1652, 1997.