A MACHINE LEARNING APPLICATION FOR FORM-FINDING OF TENSEGRITY STRUCTURES

by

Nuwan Stefan Perera

A thesis submitted to the School of Computing

In conformity with the requirements for

the degree of Master of Science

Queen's University

Kingston, Ontario, Canada

(August, 2018)

Abstract

Tensegrity (tensional integrity) is a structural principle where rigid elements (struts) under compression are held together by a network of elastic elements (cables) under tension. Tensegrity structures have many applications in modelling the natural world. Tensegrity research has been applied to fields including robotics, art, architecture, and biology. In recent years, computer simulation has been introduced as a tool to allow researchers to design, build, and simulate tensegrity structures. Structures designed both as physical models and in simulation software can require several iterations of fine adjustments.

In this thesis, we develop a form-finding application to reduce the iterative adjustments required when designing a tensegrity structure. Form-finding is the process of finding a structural configuration capable of a state of self-stressed equilibrium – when tension and compression stabilize the structure. Our form-finding application uses a tensegrity structure that is not necessarily in an equilibrium state represented as a graph as input, and produces either (a) failure when no equilibrium state is possible, or (b) a fully attributed labeled graph of a tensegrity structure in an equilibrium state. In this thesis, we use an efficient fitness function and genetic algorithms to find the stable state of a tensegrity structure. Through our form-finding application, we aim to promote the use of computer simulation, and collaboration between tensegrity researchers.

Acknowledgements

I'd like to thank Dr. Dorothea Blostein for her support throughout my thesis and thesis writing process. I would also like to thank all my family and friends who have supported me during my time and Queen's. Lastly, I would like to thank my parents for all of their support throughout my education.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Tensegrity (tensional integrity) is a structural principle where rigid elements under compression are held together by a network of elastic elements under tension. Typically, the rigid elements of a tensegrity structure are composed of wood or metal struts, while the elastic elements are composed of rubber or steel cables (Figure 1). The simultaneous forces of tension and compression within a tensegrity structure allow the structure to maintain a state of self-stressed equilibrium, making the structure both strong and flexible.

Tensegrity structures have beneficial properties including high strength-to-weight ratios, the capability to fold flat for transportation, and a high resilience to impact. The appealing properties of tensegrity structures have made tensegrity an area of interest in several domains including architecture, biology, and robotics. Section 2.3 summarizes the applications of tensegrity structures, while Section 2.4 summarizes the benefits of tensegrity structures.

The manual design and construction of tensegrity structures is a time consuming and expensive process. The design of tensegrity structures can be difficult to visualize for many researchers. Often, tensegrity structures require many fine adjustments to create a successful model. Computer simulation allows researchers to design and construct tensegrity structures in a simulated environment, and analyze a structure's interactions with its environment in a virtual setting (Section 2.6). The use of computer simulation makes design and modifications of tensegrity structures easier than when using physical models alone.

Designing efficient tensegrity structures can be a challenge in both physical and simulated environments. It is not uncommon for a researcher to design a tensegrity structure, only for it to partially or entirely collapse once exposed to simulation. In a physical setting, this can be both challenging and frustrating for tensegrity builders since the entire structure needs to be assembled again with several fine adjustments. Using computer simulation, it can also be a challenge to find an exact balance of tension and compression that keeps the tensegrity structure from collapsing. Even when a tensegrity structure does not collapse, fine tuning is often still required to obtain the desired model. For example, when designing a tensegrity model of the leg and foot, Tom Flemons spends several weeks of fine adjustments to find the right balance to produce the desired alignment of toes and ankle (Figure 2).

In this thesis, we aim to augment the design process by developing a form-finding application to find an equilibrium configuration for the tensegrity structures being simulated (Chapter 7). Our form-finding application builds upon existing form-finding methods used for membrane shell and tensegrity structures. Our application also has compatibility with widely used tensegrity simulation platforms: *The NASA Tensegrity Robotics Toolkit* and *PushMePullMe3D*. Currently neither simulation platform has the capability for automated tensegrity form-finding. Our application aims to reduce the iterative design process and fine adjustments otherwise required to simulate a tensegrity structure. By introducing a form-finding application compatible with *NTRT* and *PushMePullMe3D,* we augment the user experience when designing tensegrity structures using computer simulation.

Self-stressed equilibrium is a state where the internal tension of cables within a tensegrity structure allows all of the struts to be suspended, resulting in equilibrium of internal forces within the structure. In this thesis, we use form-finding algorithms to obtain design parameters and self-stressed equilibrium for a given tensegrity structure.

Form-finding is a technique used in structural engineering to find a particular configuration within a design space: a configuration that is in a state of equilibrium under given constraints. Form-finding is a well-established field in structure engineering, with applications to a range of structures including shell structures and tensegrity structures (Chapters 3 and 6). Within this broad research field, both physical and numerical approaches have been taken to solve form-finding. For example, a physical approach for finding the shape of a shell structure is done by suspending wet fabric and inverting that shape; these catenary shapes are materially efficient structures with a balance of tension and compression [1]. In Chapter 3, we examine numerical approaches to form-finding: Pucher's method and the force-density method summarized in Ariane Fund's thesis [1]. In Chapter 6, we examine Yamamoto's method for form-finding using genetic algorithms [2].

Currently, tensegrity simulation platforms including *NTRT* and *PushMePullMe3D* do not provide automated form-finding algorithms. For example, in *PushMePullMe3D,* a tutorial video illustrates the manual process of experimentation used for form-finding of grid shell structures (https://www.youtube.com/watch?v=K-0nHT0GeBM). Our work to add form-finding algorithms to simulation platforms shows promise for improving the design and construction process, simulation results, and adoption of computer simulation.

In this thesis we develop a platform-independent form-finding application for tensegrity structures. The application is developed with compatibility for two simulation platforms: *NTRT* and *PushMePullMe3D* (Section 2.6).

Figure 1 A 3-strut tensegrity prism showing the connectivity of struts and cables. Image from Wikimedia Commons [3].

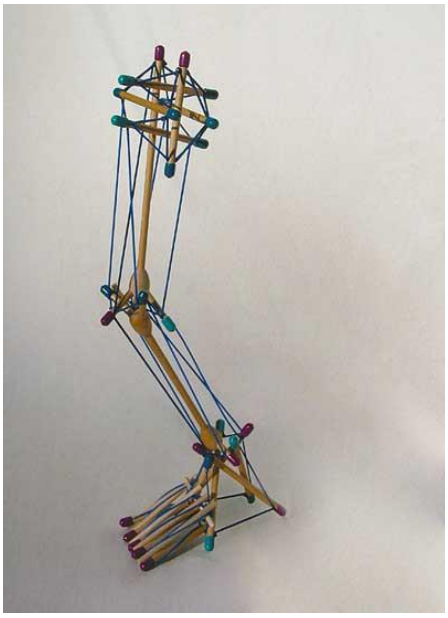

Figure 2 Tom Flemons tensegrity model of the foot and leg. © 2017 Tom Flemons, reproduced by permission.

1.1 Thesis Contributions

This thesis proposes a platform-independent tensegrity form-finding application, with extendable compatibility for *NTRT* and *PushMePullMe3D*. In the process of developing a tensegrity form-finding application, several contributions were made in this thesis.

1.1.1 Contribution 1: *Identify the need for form-finding of tensegrity structures*

Several conversations with tensegrity researchers such as Tom Flemons allowed us to identify the need for tensegrity form-finding to reduce the time and effort spent making iterative adjustments for stability. To augment the tensegrity design process, we design a stand-alone automated tensegrity form-finding application, compatible with widely used simulation platforms *NTRT* and *PushMePullMe3D* (Chapter 7). To determine fitness of tensegrity structures, our application provides generalized fitness functions for tensegrity structures. To allow customizable support for tensegrity researchers, we also provide the ability for user defined fitness functions for tensegrity structure form-finding.

Conversations with Dr. Richard Gordon (theoretical biologist and tensegrity researcher) and Tom Flemons (expert tensegrity researcher and builder) helped identify the challenges of tensegrity construction, design, and simulation. Years of email correspondence between Dr. Blostein and Tom Flemons were used to compile requirements and challenges for simulating biomechanical tensegrity structures. Skype tutorials and weekly discussions were conducted between our research group and Dr. Gordon to identify requirements and challenges for tensegrity simulation in cell biology. Based on our conversations with Tom Flemons and Dr. Gordon, we elicited a set of software requirements for simulating tensegrity models in biomechanics and cell biology.

During our Skype sessions with Dr. Gordon and attempts to replicate Flemons' models, we identified that structures can collapse in a simulated environment if there is an imbalance between tension and compression. Tensegrity structures collapsing because they are not at equilibrium also holds true for physical modelling of tensegrity structures – a common challenge faced by tensegrity builders like Tom Flemons. Often, when designing and simulating a tensegrity structure in *NTRT* or *PushMePullMe3D*, users have to iteratively adjust the structure's prestress, topology, and parameters (such as resting length for a cable) to prevent it from collapsing. By making these iterative adjustments, the structure can obtain a state of self-stressed equilibrium.

1.1.2 **Contribution 2:** ***Identify simulation support for augmenting tensegrity design***

Within the scope of this thesis, we analyze common challenges faced during the tensegrity design process by our research group and close collaborators; namely Dr. Richard Gordon and Tom Flemons. With these challenges we formulate requirements and simulation goals for our research group primarily focusing on usability, extendibility, and quality of simulation. With emphasis on these three aspects, we survey multiple state-of-the-art simulation platforms capable of supporting tensegrity structures including *NTRT* and *PushMePullMe3D* (Section 2.6). For this thesis, we identify *PushMePullMe3D* as a candidate simulation platform based on usability, extendibility, and quality of simulation . Since *PushMePullMe3D* is a proprietary software, our research group established a collaboration and an NDA/IP agreement with Dr. Gennaro Senatore, the author of *PushMePullMe3D*. Collaboration with Dr. Senatore allowed us to gain access to *PushMePullMe3D* source code to build upon the requirements of Tom Flemons and Dr. Richard Gordon for tensegrity modelling.

1.1.3 **Contribution 3:** ***Survey state-of-the-art form-finding algorithms***

Form-finding was found as a solution for obtaining a structure capable of self-stressed equilibrium. After identifying form-finding as a solution to one of our problems a literature review of state-of-the-art form-finding algorithms and applications to tensegrity structures was conducted (Chapters 4 and 6). By reviewing literature on form-finding, we were able to identify suitable methods for developing a form-finding framework to support the tensegrity design process.

1.1.4 **Contribution 4:** ***Prototype implementation of a form-finding application***

To create a framework for tensegrity form-finding we extensively studied the works of Fund (2008) and Yamamoto (2011) [1, 2]. Fund analyzes existing numerical approaches to form-finding of membrane shell structures, with many concepts that are applicable to tensegrity structures (Chapter 3). Yamamoto uses genetic algorithms coupled with the force-density method to address form-finding specifically for tensegrity structures (Chapter 6). During this thesis research, both these methods were carefully implemented in MATLAB to further understand the mechanics of these approaches, the results of these investigations can be found in Chapters 3 and 6.

In this thesis we propose a platform-independent genetic algorithm based form-finding application for tensegrity structures. Our proposed approach expands on the ideas from Fund and Yamamoto's works as well as other state of the art approaches to tensegrity form-finding (Chapter 7). This contribution also defines a representation of tensegrity structures new to our research group, based on approaches from the literature (Chapter 4). Together, the proposed form-finding application and tensegrity structure representation serve as a major contribution in

this thesis, promoting collaboration and adoption of tensegrity simulation software in the field of tensegrity research.

Chapter 2

Tensegrity Structures: Definition, History, and Applications

Tensegrity structures were first used in art and architecture as early as the 1960s by Buckminister Fuller and Kenneth Snelson. The field of tensegrity has since grown, as it has been applied to domains including robotics, biology, and structural mechanics. This chapter provides background on the definition, benefits, and practical applications of tensegrity structures.

2.1 Defining a Tensegrity Structure: Assumptions and Constraints

Tensegrity is a structural principle where a network of elastic elements under tension suspend compressed rigid elements. Tensegrity structures exhibit a flexible and resilient balance of tension and compression by obtaining a state of self-stressed equilibrium. When an outside force is applied to a tensegrity structure, the entire structure deforms slightly as it moves into a new state of equilibrium; this global response makes tensegrity structures highly resistant to impact. Typically tensegrity structures are constructed using metal or wood struts connected by steel or rubber cables.

Tensegrity structures obtain their stability through the continuous network of tension counteracted by the discontinuous compression. The tension and compression of a tensegrity structure is clearly visible as struts float in a network of cables; this is sometimes referred to as *floating compression* [4]. In a pure tensegrity structure, struts are not allowed to be in contact with each other, and each node is connected to exactly one strut [5, 6]. Skelton and de Oliveira refer to pure tensegrity structures as *class 1* tensegrity structures [6]. Sometimes, pure tensegrity structures are not sufficient for the desired application. Increasing the number of struts connected at each node can be a solution to overcome these challenges. If a tensegrity structure has two

struts connected to some nodes, it is referred to as a *class 2* tensegrity structure. This typology continues for *class 3* tensegrity structures, where three struts can be connected at a node [6]. In this thesis, we consider *class 1* tensegrity structures for simplicity.

Tensegrity structures are supported by a balance of their internal tensile and compressive forces, making them *self-stressed*. A valid tensegrity structure is balanced by its self-sustaining internal forces, without reliance on external supports or forces. When a tensegrity structure is balanced by its internal forces, it is known to be in a state of *self-stressed equilibrium* [2, 7].

Tensegrity structures consist of a network of continuous tension complemented by discontinuous compression. The continuous tension is made up by the network of cables within the structure. This network should be continuous with each node connected to two or more cables. The struts of a tensegrity structure should be discontinuous, suspended by the tension of the cables. With a continuous network of cables suspending the struts, the struts of a tensegrity structure should not touch each other when at rest.

In this thesis, we place the following constraints on defining a valid tensegrity structure. These criteria are met by all structures that can be discovered by our form-finding method.

- *Class 1 Tensegrity Structure:* Each node can be connected to exactly one strut, known as a *class 1* tensegrity structure. Thus, strut-strut connections are assumed to be invalid in our application. By considering only *class 1* tensegrity, we eliminate boundary constraints required for form-finding of higher class tensegrity structures with multiple struts connected at a node [6].

- *Struts are infinitely rigid:* Struts cannot be deformed or bent regardless of load exerted. This constraint is placed on tensegrity structures in our application for simplicity in the methods used for form-finding.

2.2 Tensegrity Structures as a Graph

In this thesis, we describe tensegrity structures using terminology from graph theory. Each connective element (strut or cable) of the tensegrity structure is represented as an edge, while each connection point is represented as a node. Edges are labeled as *strut* or *cable*, corresponding to their physical properties. Edge attributes provide information about material properties. For example, each edge can have attributes *length*, *stiffness*, and *radius*. Node attributes provide spatial location information such as $x, y,$ and z coordinates in 3-space. Figure 3 illustrates the representation of a tensegrity structure as a graph.

In this thesis, form-finding takes a partially-unattributed graph as input, producing as output either (a) failure when no equilibrium state is possible, or (b) a fully attributed labeled graph of a tensegrity structure in an equilibrium state. Form-finding methods often differ in exact input and output: some methods do not require length attributes for edges or spatial coordinates for nodes, while these attributes are essential in other methods. Chapters 3 and 6 provide more details on form-finding methods.

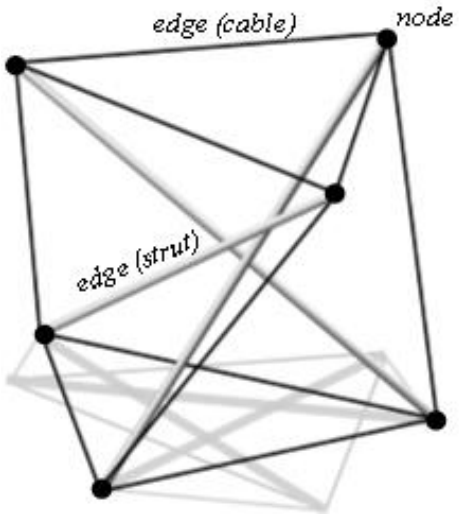

Figure 3 Representation of a tensegrity structure as a graph showing nodes and edges in 3D. Figure adopted from Wikimedia Commons and annotated by Nuwan Perera [8].

2.3 Applications of Tensegrity

Tensegrity research has been applied to a number of research disciplines including art, architecture, biology and robotics.

Kenneth Snelson, a tensegrity pioneer and artist, was influential in introducing tensegrity as an art form. Tensegrity has been used in sculptures showing the balance of tension and compression within structures. Many artistic applications of tensegrity are coupled with architectural and structural applications of tensegrity [9].

Tensegrity structures have been used in a number of structural engineering and architecture applications. The resiliency of tensegrity structures makes them appealing for structures that are subject to environmental stresses such as wind and earthquakes [9]. Some applications of tensegrity structures include bridges, signal towers, and dome roofs (Figure 4 and 5). For example, the Kurilpa Bridge in Brisbane, Australia is a tensegrity based bridge (Figure 5).

These applications of tensegrity continue to be investigated for adaptive structure and structural mechanics research.

In biology, many researchers have investigated the use of tensegrity to model biological structures from anatomical structures to cellular structures [10]. Researchers such as Tom Flemons have investigated tensegrity for modelling anatomical structures and the interactions between connective tissue and bones within the human body (Figure 6) [11, 12]. In microbiology, tensegrity structures have been used to model the cytoskeleton, the structural framework of a cell, as well as structures within the cell by researchers such as Dr. Richard Gordon and Dr. Donald Ingber [13, 14]. The use of tensegrity structures in biology shows potential to aid in understanding mechanical interactions within anatomical structures and cells, and their effects in sports injuries and diseases respectively.

In robotics, tensegrity research has been prominent for space exploration robotics. Research by *NASA* has shown promise in the use of tensegrity robotics for space exploration using the *Superball* (Figure 7). Tensegrity robotics research has also made contributions to tensegrity simulation through the *NASA Tensegrity Robotics Toolkit* and machine learning for tensegrity control [15, 16]. Tensegrity is appealing to robotics research because of its resiliency, ability to fold easy for transportation, and energy efficiency by efficient use of materials [11, 15, 17].

Since tensegrity research is used in a number of application domains, there is promise for interdisciplinary collaboration to improve the quality of tensegrity simulation. Promoting collaboration and accessibility to tensegrity simulation will enable advancement in the field of tensegrity research.

Figure 4 Tensegrity Needle Tower by Kenneth Snelson. Photo by BAR Photography, Tensegrity Wiki [18].

Figure 5 Kurilpa tensegrity bridge in Brisbane Australia. Image from Wikimedia Commons [19].

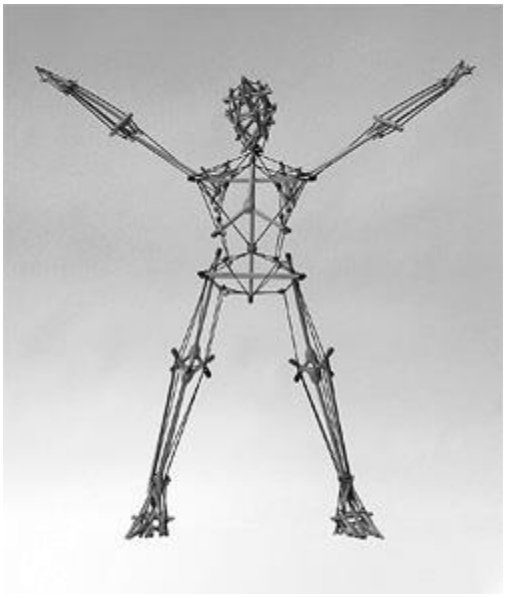

Figure 6 Tom Flemons full body tensegrity model. © 2017 Tom Flemons, reproduced by permission.

Figure 7 NASA Superball tensegrity robot. Image from Wikimedia Commons [20].

2.4 Benefits of Tensegrity Structures

Tensegrity structures have many characteristics that make them an appealing paradigm for modelling structures in fields such as biology, robotics, art, and architecture. Many conventional models built by humans place emphasis on compression and do not focus on the tensile elements of a structure [17]. For example, a brick wall is mostly constructed using the compression forces of bricks with little to no tension within the structure. In the natural world, compressive forces are often counterbalanced by tensile components. For example, fascial tissue (tension) interacts with the bones (compression) of the human body [10, 11, 21].

The balance of tension and compression within a tensegrity structure provides appealing properties for applications in many research areas. Some benefits of tensegrity structures are described below.

- *Resiliency:* Tensegrity structures have high structural resiliency, allowing the structure to absorb impact, and return to a state of equilibrium [11]. When external forces are exerted on a tensegrity structure, the structure can propagate forces through its tensile network preventing permanent deformation. If a tensegrity structure is subject to local deformation, although slightly degraded, it can still maintain function [5]. This degree of structural resiliency is appealing in many structural engineering and robotic applications when tensegrity structures are subject to impacts causing deformation; in contrast, rigid structures often cannot return to an equilibrium state.
- *Fold flat for transportation:* Unlike stiff structures, compressive elements of a tensegrity structure are discontinuous allowing the structure to be folded flat for transportation [11, 17, 22]. The ability to fold flat for transportation and return to a state of self-stress equilibrium is an appealing property for applications where transportation costs can be expensive such as space robotics.

- *Light-weight:* Tensegrity structures are composed of strut and cable elements. These elements are often lighter weight than fully rigid structures [11, 22].
- *Easy to simulate:* Compared to other structures like fluids, tensegrity structures are easier to simulate due to the mechanics of struts and cables. Furthermore, assuming that only axial forces occur within a tensegrity structure and struts cannot bend, simplifies the simulation process for tensegrity structures without compromising the integrity of the simulation [17, 22].

The appealing properties of tensegrity structures creates interest from many domains, resulting in a collaborative and interdisciplinary field.

2.5 Manual Construction of Tensegrity Structures

Tensegrity structures are conventionally designed and assembled by manually connecting struts and cables; this is often done by a tensegrity building expert. Even for a simple tensegrity structure such as the 3-strut prism, a careful assembly process is required since all components of the structure must be connected together before the desired state of self-stress is obtainable. Most tensegrity structures used in research are intricate, and require many iterations of adjustments before receiving a desired outcome. Thus, designing and constructing tensegrity structures suitable for research applications can be challenging; especially for designers new to tensegrity research. To augment the process of manual tensegrity construction, some researchers have created computer simulation tools for tensegrity modelling.

2.6 Computer Simulation of Tensegrity Structures

Computer simulation of tensegrity structures aims to address the challenges faced by researchers during manual construction. The use of computer simulation aims to allow researchers to develop structures quickly, increase collaboration, and produce precise quantitative results.

Computer simulation allows researchers to design and simulate rapid prototypes of tensegrity structures with reduced effort and cost compared to physical model building. In a physical model, any change in the connectivity of a tensegrity structure could require the entire structure to be disassembled and reconstructed. This process diverts time and effort away from conducting experiments with their structures. Simulation makes connectivity changes easier, because structures can be modified within the simulation engine. Once a desired structure is designed, the structure can be simulated to see whether or not it behaves as the researcher expects. If the structure meets the design criteria, it can then be built as a physical tensegrity model for further experimentation.

In addition to physical modelling, computer simulation fosters collaboration between tensegrity researchers. Unlike physical models which can take a long time to replicate and share, computer simulation allows tensegrity structures to be stored as data files which can be easily replicated and shared among tensegrity researchers. By promoting collaboration, increased adoption of computer simulation is a key to advancing the field of tensegrity research.

Unlike physical modelling, the use of physics in computer simulation means that precise values are computed throughout the simulation. Values such as tension within a cable can be difficult to obtain from a physical model, but are easy to access in a simulation engine. The use of

computer simulation can allow researchers to report more quantitative metrics on tensegrity structures by exploiting the calculations made by the physics engine. Ensuring that physics simulation properly captures the mechanical behavior of a physical model can be established through validation procedures such as those reported by Caluwaerts et al [15].

This thesis work has examined two simulation platforms capable of simulating tensegrity structures *The NASA Tensegrity Robotics Toolkit (NTRT)* and *PushMePullMe3D*.

2.6.1 The NASA Tensegrity Robotics Toolkit

The *NASA Tensegrity Robotics Toolkit (NTRT)* is an open-source platform used for simulating tensegrity structures, with an emphasis on tensegrity for space robotics. *NTRT* is compatible with Linux and employs the Bullet Physics Engine for simulation.

Currently, *NTRT* does not support a graphical user interface for installation or designing tensegrity structures. In order to use *NTRT* coding knowledge is required; shell scripting is required to install the platform and knowledge of C++ is required to create tensegrity structures. This is a challenge for many tensegrity researchers coming from backgrounds that do not possess programming skills.

NTRT is developed for simulating tensegrity robotics; placing constraints on the simulation engine for robotic structures [16]. Therefore, the physics simulation used by *NTRT* can be a limitation for researchers applying tensegrity to other domains such as biology.

Although *NTRT* can be a limitation for some researchers and research domains, the simulation platform has a large user-base focused on tensegrity research. The prominent user-

base makes *NTRT* a valuable resource for tensegrity researchers when using and developing simulation for tensegrity. Overall, *NTRT* provides researchers with an efficient simulation platform for conducting tensegrity research for space robotics.

2.6.2 PushMePullMe3D

PushMePullMe3D is a simulation platform developed in *Java* by mechanical engineer Gennaro Senatore. *PushMePullMe3D* is available as a part of *Expedition Workshed*, an online collection of learning materials for engineering students, and teachers. *PushMePullMe3D* is free to use available as a *Java Network Launch Protocol (JNLP)* application. However, the source code for *PushMePullMe3D* is proprietary and Queen's University created an NDA/IP agreements in order for us to gain access. Although *PushMePullMe3D* is a platform targeted at education, it is a valuable simulation tool for many tensegrity researchers.

The proprietary code of *PushMePullMe3D* has led to a clean software design and architecture, making future extensions easier to facilitate. This can be attributed to the limited number of developers working on *PushMePullMe3D*. However, with the limited number of developers, obtaining development support can be more challenging than open-source platforms like *NTRT*.

The user interface of *PushMePullMe3D* allows researchers to design and adjust tensegrity structures in a graphical user interface (GUI). The intuitive GUI is a beneficial tool for researchers with limited programming knowledge, making them more comfortable and likely to adopt computer simulation.

As a contribution to this thesis, we identify *PushMePullMe3D* as a candidate platform for carrying out future tensegrity research and established a research agreement to access the platform's source code. By identifying *PushMePullMe3D* as a candidate platform, this thesis has laid the groundwork for future extension to *PushMePullMe3D* as a tensegrity research simulation engine.

Chapter 3

Form-Finding Structures

Humans have been using intuition and experimentation to develop efficient and optimized structures for many centuries. Form-finding formalizes the development of efficient and optimized structures using algorithms and numerical methods [1]. Form-finding is a technique used in structural engineering for finding a structural configuration that is in an equilibrium shape [1, 2, 7]. Although network, membrane shell, and tensegrity structures present themselves very differently, many of the underlying concepts from their form-finding methods are transferrable between these types of structures. This chapter provides an overview of form-finding structures, specifically looking at form-finding of membrane shell structures, a topic covered extensively by Ariane Fund [1].

3.1 Form-Finding of Membrane Shell Structures

This section outlines form-finding of membrane shell structures. A membrane shell structure is a curved shell with small thickness relative to its other dimensions. Ideally a membrane shell structure is capable of supporting both tensile and compressive loads [23]. Shell structures can be seen in the construction of large dome roofs used for stadiums or airports [1].

Heinz Isler pioneered the thin-shell concrete construction by studying the catenary behavior of suspended fabric. A catenary curve is the shape assumed when a cable is hanging under its own weight, fixed between two points. Isler performed the design process by hanging wet fabric and letting it freeze; or soaking fabric in cement mix and letting it dry. The sagged form results in a catenary curve by the tension under its own weight; when inverted, the sheet forms a structural shell in compression [1].

The goal of form-finding is to automate Isler's process and other similar processes for structure design. Instead of hanging fabric, form-finding aims to use an algorithm to find the shape of the structure. To find the desired shape, the user supplies a known external load that is to be applied to the shell. The desired shape has been achieved when the structure is rigid and there are no bending or shear forces, only tension and compression [1]. The final shape is determined by the specified external load, changing based on the change in load.

Ariane Fund's thesis examines the difference between two form-finding methods for membrane shell structures: Pucher's method and the force-density method. Fund creates a MATLAB based form-finding program to determine an equilibrium shape of a membrane shell structure based on the location of external supports and a specified stress pattern [1].

3.1.1 Pucher's Method

Pucher's method uses Pucher's equation to perform form-finding of membrane shell structures. Pucher's equation reduces multiple force-equilibrium equations into one differential equation [1, 24]. Pucher's equation is expressed as a function of membrane stress and elevation at each point [1].

For Pucher's method, nodes are labelled as fixed nodes and internal nodes. Fixed nodes are specified by $x, y,$ and z so they cannot be moved in any direction. Internal nodes are specified by only x and y coordinates, allowing displacement in the z direction. The method uses finite element analysis and triangular elements to determine the z coordinates for each internal node when loads are applied in a vertical direction [1].

3.1.2 Force-Density Method

The force-density method for form-finding uses the force-to-length ratio at each edge of the tensegrity structure to find a state of equilibrium [1, 7]. Using the force-density method, the structure is at equilibrium when the sum of all forces at each node is equal to zero [7].

The force-density method uses a 2-dimensional topology graph representing a pin-jointed network of edges in a plane (Figure 8). Along with the topology graph, a list of force-densities at each edge, and the x, y and z coordinates of each fixed node of the structure are also given as input. Each free node has an external load defined in 3-dimensional space. The output of the force-density method consists of x, y and z coordinates for each free node in the structure and the forces at each edge. Figure 8 illustrates the results of the force density method on a membrane shell structure [1].

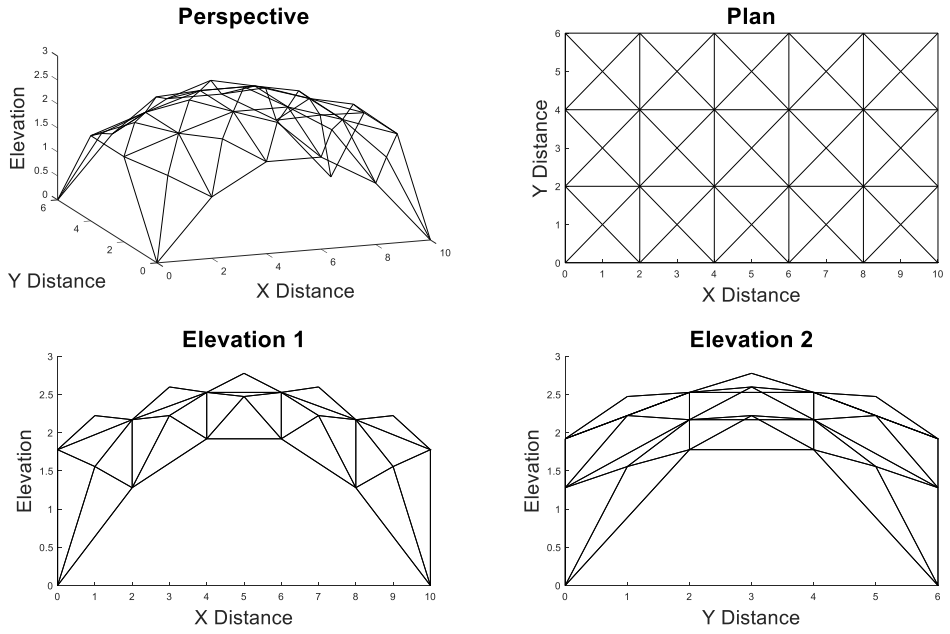

Figure 8 Sample form-finding of a membrane shell structure using the force density method implemented by the MATLAB code in Fund's thesis. *Perspective*, *Elevation 1*, and *Elevation 2* show different angles of the output shell structure. *Plan* shows the topology graph input of the shell structure. The four corners of the membrane shell structure are fixed nodes anchored to the ground producing an inverted catenary curve. Figure 4-4 from [1]: Reproduced with permission from Ariane Fund.

3.2 Application of Form-Finding to Tensegrity

Form-finding of shell structures is an established area of research, providing useful insights for applying form-finding to tensegrity structures. Tensegrity structures and shell structures share many similarities such as developing internal forces of both tension and compression. However, there exists an important difference: shell structures require external supports, while tensegrity structures are supported by a state of self-stressed equilibrium. This section looks at Fund's work on form-finding of shell structures with potential ways to adapt these methods for tensegrity structures. As described below, many methods in the field of

tensegrity form-finding are influenced by methods from form-finding for membrane shell structures.

Based on Fund's thesis, Pucher's method would not be suitable for applications to tensegrity form-finding [1]. In Pucher's method, some fixed nodes are required, and free nodes must be fixed in the x and y directions with movement only in the z direction during form-finding. This is not suitable for tensegrity structures where all nodes are free and can be moved in any direction.

The force-density method can be adapted for tensegrity form-finding. When applied to a shell-structure, the force density method uses a combination of fixed and free nodes. To adapt the force-density method for tensegrity structures, all nodes must be free. Unlike Pucher's method, the force-density method allows coordinates of free nodes to be moved in the $x, y,$ and z directions [1].

Several previous researchers have applied the force-density method to tensegrity form-finding [2, 25, 26, 27]. The form-finding methods proposed in the literature have not been made available in widely used simulation platforms, where researchers can use these advancements in tensegrity research. Currently both *NTRT* and *PushMePullMe3D* do not have the ability for automated form-finding. This thesis gives tensegrity researchers access to automated form-finding capabilities compatible with *PushMePullMe3D* and planned compatibility with *NTRT*.

In other approaches to tensegrity form-finding, the force-density method has also been coupled with machine learning approaches such as Monte Carlo simulations and genetic algorithms.

Li et al perform Monte Carlo simulation for tensegrity form-finding [28]. This method has been successful; however, Monte Carlo simulation is often a computationally expensive process. Li et al capitalize on the effectiveness of Monte Carlo for large tensegrity structures [28].

Many researchers use genetic algorithms and evolutionary computing for tensegrity form-finding [2, 26, 29, 30, 31, 32]. Often genetic algorithm approaches to tensegrity form-finding focus on a particular type of tensegrity structure. For example, both Yamamoto et al and Gan et al focus on the form-finding of irregular tensegrity structures (tensegrity structures with asymmetric topologies) [2, 28, 31]. Using genetic algorithms, researchers including Yamamoto et al are faced with difficulties to guarantee convergence [2]. Although genetic algorithms can be challenging to implement for tensegrity form-finding, they often require less computational time than Monte Carlo simulation. Since most tensegrity structures taken as input are assumed to be close to a stable state, the genetic algorithm should not be a computationally expensive process for the user. In this thesis, we develop a form-finding approach for tensegrity structures using genetic algorithms (Chapter 7). Chapter 5 summarizes genetic algorithms, while Chapter 6 summarizes Yamamoto's approach to tensegrity form-finding using genetic algorithms.

Providing widely-available implementations of these form-finding methods would allow researchers to exploit the advances in tensegrity design automation. This thesis provides tensegrity researchers with automated form-finding capabilities compatible with *PushMePullMe3D* and planned compatibility with *NTRT*.

Chapter 4

Representation of Tensegrity Structures

Our tensegrity form-finding application must represent tensegrity structures in a human readable and computationally efficient format. Tensegrity structures have different representations across platforms and disciplines within the literature. The differences in representation of tensegrity structures is a challenge for collaboration and validation of methods between researchers. This chapter summarizes mathematical and computational representations of tensegrity structures.

A tensegrity structure can be described as an undirected graph. Each edge of the graph is labeled as a *strut* or *cable.* Each edge has attributes such as *length*, *stiffness*, and *radius.* For mathematical benefits in form-finding approaches, each edge of a tensegrity structure can be given a directionality, to create a directed graph [2, 7, 25, 26, 31]. Section 4.1 describes a topology graph representation of tensegrity structures. Section 4.2 describes a mathematical representation of tensegrity structures.

4.1 Non-planar Topology Graphs for Tensegrity Structures

The connectivity of a tensegrity structure can be represented as a topology graph. Ariane Fund's thesis illustrates the use of topology graphs for the abstract representation of membrane shell structures (Figure 9) [1]. With minor modifications, Fund's representation of topology graphs can be used for the abstract representation of tensegrity structures.

Fund represents the topology graph of a membrane shell structure as the connectivity in the x-y plane. For membrane shell structures, all node inputs (fixed and free nodes) for form-finding are specified with x and y coordinates, while the form-finding solves for z coordinates of free nodes [1, 7]. For membrane shell structures, the topology graph is constrained such that two edges cannot intersect – so edges connect adjacent nodes vertically, horizontally or diagonally. This provides an aerial view of the structure to represent its connectivity.

For topology graphs to effectively represent tensegrity structures, the graphs must be non-planar allowing intersections in the x-y plane. The reason that planar topology graphs suffice for shell membrane structures is because the shell structures have additional constraints provided for fixed nodes, preventing overlap in edges. Tensegrity structures do not have fixed nodes, instead they require enough edges to create a full 3-dimensional surface shape. This forms a non-planar graph when projected onto the x-y plane.

Topology graphs provide researchers with an abstract representation of tensegrity structures during the design and construction process. However, a different computational representation is needed to support the mathematical operations that form-finding algorithms perform on the coordinates, nodes, and edges of a tensegrity structure. Section 4.2 outlines the mathematical representation of a tensegrity structure, a representation which can be derived from the topology graph shown in this section. In this thesis, we choose to represent the graph as a matrix-vector combination based on literature surveyed in the field of tensegrity form-finding [2, 25, 26, 31].

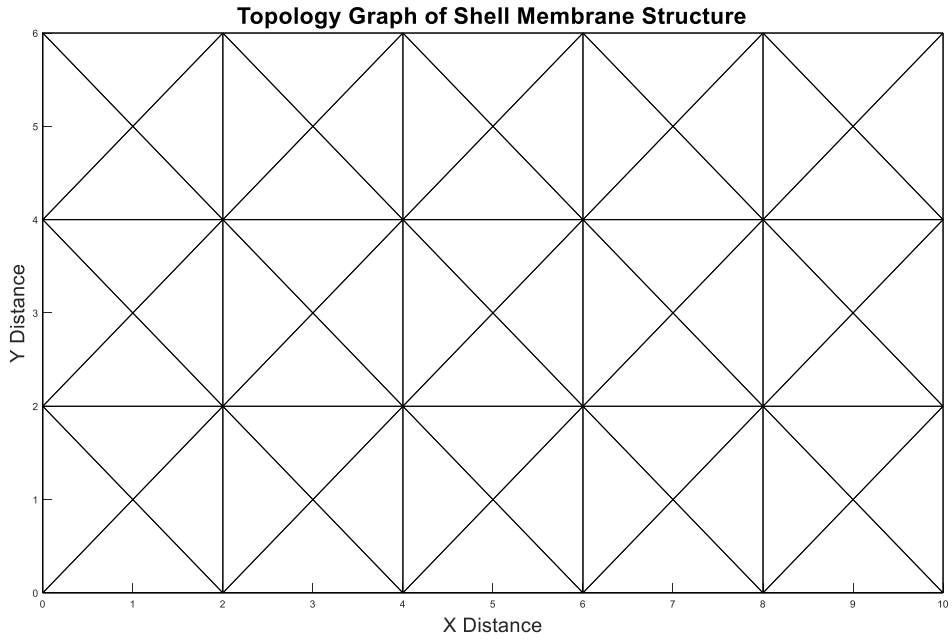

Figure 9 Planar topology graph used as input for form-finding of membrane shell structures in Fund's thesis. Reproduced with permission from Ariane Fund [1].

4.2 Mathematical Representation of a Tensegrity Structure

For tensegrity form-finding methods, including the force-density method, it is important for tensegrity structures to be represented mathematically. The mathematical representation allows computations such as vector and matrix multiplication to be carried out to determine characteristics of the tensegrity structure being examined. These computations are essential in the tensegrity form-finding process. This section outlines the notation and mathematical representation of tensegrity structures. The mathematical notation used in this thesis is a common notation with influence from notation used by several papers including those by Yamamoto, Gan, Koohestani, and Estrada [2, 25, 26, 31]. Developing a common notation used in this thesis required careful consideration to the specifications of different tensegrity form-finding algorithms.

Mathematically, a tensegrity structure can be defined using a matrix-vector combination. As described in more detail below, the matrix represents each edge of the tensegrity structure and the connectivity of the edge between two nodes. The vector attributes each edge with a type, strut or cable. Together the matrix-vector combination is able to define a tensegrity structure. This approach can be seen in many tensegrity form-finding approaches [2, 17, 31, 30, 33].

The connectivity matrix, C, describes the connectivity of a tensegrity structure, the connection of nodes by each edge. In a generalized form, tensegrity structures have n nodes and b edges. In the connectivity matrix, each row represents an edge, while each column represents a node. So the connectivity matrix is of size b by n. The connectivity contains -1 and 1 to indicate nodes connected at an edge and its directionality (Figure 10). Tensegrity structures are represented in this way as directed graphs because of the computations required in the force-density method for form-finding [7]. Directionality of a tensegrity graph can be assigned arbitrarily, such as going from -1 to 1 in a clockwise direction.

The force-density (also referred to as tension coefficient) vector, $\vec{q}$, describes the type of each edge of the tensegrity structure, strut or cable. The vector labels each edge with a tension coefficient, describing the tension at a given edge. For struts, which are rigid, the force-density value is labelled -1 [2]. Cables are given force-densities greater than 0, although some methods assume force-densities of 1 for all cables [2, 31].

Figure 11 provides an example of a connectivity matrix and force-density vector. Based on the matrix-vector combination, it is possible to estimate nodal coordinates for a tensegrity structure [7, 25].

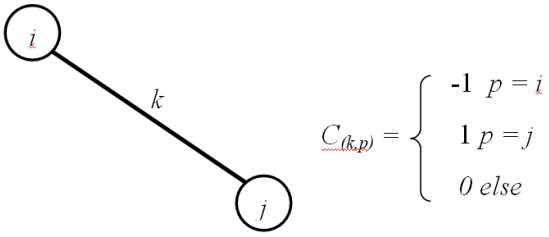

Figure 10 Example of an edge connected by two nodes and its representation in a connectivity matrix. Figure created by Nuwan Perera adopted by work from Gan et al [31].

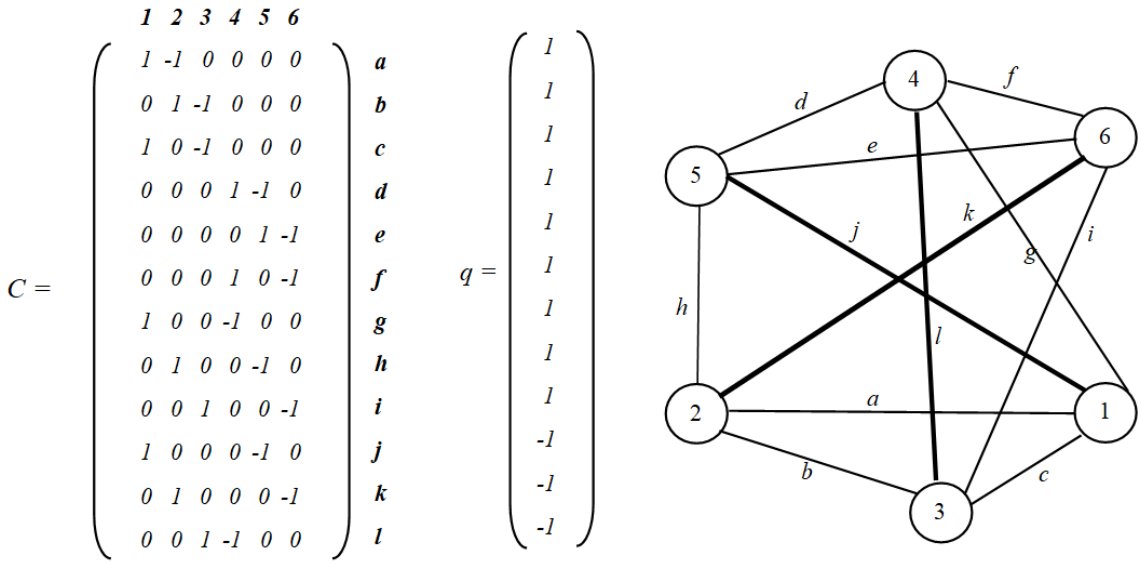

Figure 11 Connectivity matrix and tension coefficient vector representing a tensegrity 3-prism (right). Figure created by Nuwan Perera adopted from work by Yamamoto et al [2].

Symbol	Description	Type
b	Number of edges	Integer
n	Number of nodes	Integer
d	Dimensionality of tensegrity structure	$d = 3$
C	Connectivity matrix where element k is connected between nodes i and j by -1 to 1.	Matrix of size $b \times n$
$\vec{q}$	Tension coefficient vector	Vector of size $b \times 1$

Table 1 Summary of notation for representing a tensegrity structure.

Chapter 5

Genetic Algorithms

Genetic algorithms are a class of evolutionary algorithms inspired by biological processes in evolution and genetics [34]. Genetic algorithms are an effective solution for optimization problems with a large search space. This makes the use of genetic algorithms well-suited for form-finding of tensegrity structures, where the search space greatly increases with the number of nodes.

This chapter provides a general review of genetic algorithms as background for the Chapter 6 summary of existing work in tensegrity form-finding using genetic algorithms, and the approach to form-finding developed in this thesis in Chapter 7.

The operations of a genetic algorithm are inspired by biological concepts from natural selection. In a genetic algorithm, a population of individuals are created, where only the 'fittest' individuals from the population are selected to the next generation. The next generation is generated by genetic operations aiming to inherit the best traits from its parent generation. This process continues until an individual or multiple individuals satisfy the fitness measure specified, a local optimum in the search space.

Genetic algorithms typically have three key operations known as genetic operators: mutation, crossover and selection [34]. In genetic algorithms, the genetic operators (mutation, selection, and crossover) are applied to the population to create diverse individuals, similar to genetics in biology [35].

- **Mutation** is a process in which an individual's gene(s) are randomly altered with intent of perhaps increasing its fitness (Figure 12). Mutations can be either advantageous or disadvantageous to an individual; changing its characteristics from its parent [34].
- **Crossover** is a process in which two individuals of a previous generation are 'hybridized' to create a new individual analogous to biological reproduction (Figure 13). The goal of crossover is to create a more 'fit' individual by taking the best qualities in its 'parents' [34].
- **Selection** is a process in which the individuals of the population are compared against some fitness function. *Fitness functions* provide feedback to the genetic algorithm on the favorability of an individual and its proximity to the optimum or goal state. The 'fittest' individuals, being those who score highest according to the fitness function, are then selected to be used in the next generation [34].

To apply genetic operators, it is important that the data is represented as a chromosome encoded in a vector or string. Often the data used for a genetic algorithm is not natively represented as a vector or string, so a conversion or *encoding* needs to occur. The conversion of data into a chromosome is known as chromosome encoding. This allows genetic operators, specifically mutation and crossover, to be applied to the data creating diversity in the population.

Overall, genetic algorithms are an efficient technique for solving optimization problems in a large search space such as tensegrity form-finding [2, 26, 31, 33].

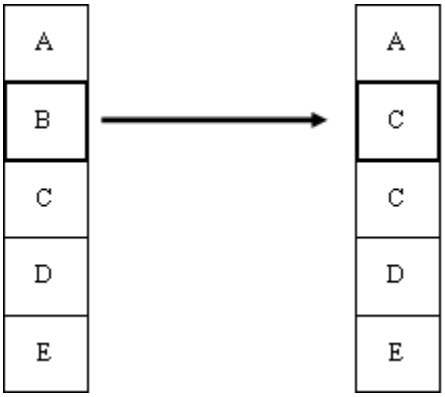

Figure 12 Example of mutation operation of genetic algorithms. Figure created by Nuwan Perera.

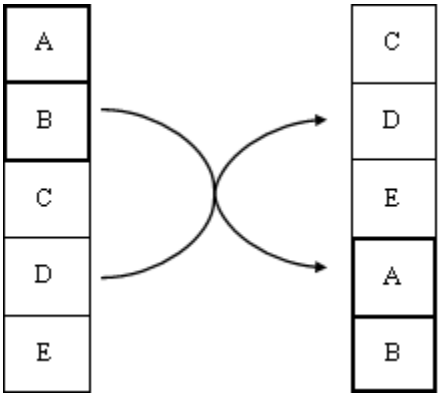

Figure 13 Example of crossover operation of genetic algorithms. Figure created by Nuwan Perera.

Chapter 6

Yamamoto's Form-Finding Using Genetic Algorithms

This chapter outlines the use of genetic algorithms for tensegrity form-finding based on work by Yamamoto et al. To understand state of the art in tensegrity form-finding, we conducted an extensive review and implementation of work by Yamamoto et al [2]. This chapter reviews the work done by Yamamoto et al, which is reimplemented and further extended in our form-finding approach in Chapter 7.

Yamamoto's method uses the force-density method with genetic algorithms for form-finding of tensegrity structures. As a contribution in this thesis, a detailed analysis and implementation of Yamamoto et al's method was performed to understand the state of the art in tensegrity form-finding using genetic algorithms. Within the same research group, Gan et al extend the work of Yamamoto, providing many clarifications to the process of tensegrity form-finding and improving the fitness measures used in the original paper. This chapter outlines the genetic algorithm approach to tensegrity form-finding proposed by Yamamoto et al (2011) and extended by Gan et al (2015) [2, 31].

In both these papers, form-finding is defined as the process of finding a structural configuration capable of a state of self-stressed equilibrium. Self-stress equilibrium is obtained when all struts are suspended by the cables of the structure, creating an internal state of equilibrium between tension and compression forces [31, 36]. Both the methods by Yamamoto and Gan perform form-finding on an attributed tensegrity structure. Yamamoto and Gan both use genetic algorithms during the form-finding process to adjust strut length, cable length, and connectivity of the structure to obtain a configuration for self-stressed equilibrium. If such a

configuration exists, then both methods report success and produce an attributed tensegrity structure with x, y and z coordinates describing a state of self-stressed equilibrium [2, 31].

These methods have constraints and assumptions for the tensegrity form-finding problem. In Yamamoto's method, it is assumed that all struts are denoted by a tension coefficient of -1 while all cables are denoted by a tension coefficient of 1 [2]. This means that all cables share the same mechanical properties, and all struts share the same mechanical properties.

Further sections of this chapter outline the mechanics of Yamamoto's form-finding method using genetic algorithms.

6.1 Algorithm Input

Yamamoto's method takes an attributed tensegrity structure as input. As mentioned above, Yamamoto uses genetic algorithms during the form-finding process to adjust strut length, cable length, and the connectivity of the structure to obtain a configuration capable of self-stressed equilibrium.

As described in Chapter 4, the attributed tensegrity structure is represented by the connectivity matrix C and tension coefficient vector $\vec{t}$. The connectivity matrix represents the connectivity of edges between nodes. The tension coefficient vector represents the tension for each edge in the tensegrity structure. Using these inputs, Yamamoto's method is capable of determining x, y, and z coordinates of the nodes in a tensegrity structure.

6.2 Fitness Functions

The use of fitness functions to determine whether or not a tensegrity structure is in a state of self-stressed equilibrium is essential for form-finding using genetic algorithms. For tensegrity form-finding, fitness functions determine how close the tensegrity structure is to a state of self-stressed equilibrium.

To obtain a robust fitness function, Yamamoto uses the sum of multiple fitness functions [2]. By using multiple fitness functions, independent aspects of the tensegrity structure can be taken into account to analyze the overall fitness of the structure.

$$f_{total} = f_1 + f_2 + f_3$$

This chapter uses notation consistent with that presented in Chapter 4. A summary of notation is provided by Table 1 in Chapter 4.

6.2.1 Fitness Function 1: *Self-Stressed Equilibrium*

The first fitness function, f_1, determines whether or not the tensegrity structure is capable of a state of self-stressed equilibrium. The use of self-stressed equilibrium is a well-established method for tensegrity form-finding stemming from the force-density method proposed by Schek in the 1970s for mesh and shell structures, and further applied to tensegrity structures by several other researchers [2, 7, 17, 25, 30, 33, 37].

Using C and $\vec{q}$, Yamamoto computes the force-density matrix D. The force-density matrix represents the force-density values throughout the tensegrity structure [7, 37, 38].

$$D = C^T diag(\vec{q}) C$$

The force-density matrix can be used to approximate 3-dimensional node coordinates for the tensegrity structure using Schur decomposition [2, 25, 31]. The first three eigenvectors of matrix U represent the x, y, and z coordinates of the tensegrity structure in 3-dimensional space.

$$D = U^T V U$$

$$[\vec{x}\ \vec{y}\ \vec{z}] = [\overrightarrow{\boldsymbol{u_1}}\ \overrightarrow{\boldsymbol{u_2}}\ \overrightarrow{\boldsymbol{u_3}}]$$

Using the 3-dimensional coordinates and connectivity matrix of the tensegrity structure, Yamamoto computes the equilibrium matrix A. The equilibrium matrix provides insight into whether or not the tensegrity structure being examined is capable of a state of self-stressed equilibrium.

$$A = \begin{bmatrix} C^T diag(C\vec{x}) \\ C^T diag(C\vec{y}) \\ C^T diag(C\vec{z}) \end{bmatrix}$$

When the singular tension coefficients of A approach zero, the structure approaches a state of self-stressed equilibrium. Yamamoto computes the singular tension coefficients by applying singular value decomposition to the equilibrium matrix.

$$SVD(A) = UVW^T$$

The bottom right corner of matrix V determines whether or not the tensegrity structure is in a state of self-stressed equilibrium, fitness value f_1 [2, 31].

$$f_1 = V_{end,end}$$

6.2.2 Fitness Function 2: *Connectivity at a node*

For a tensegrity structure to be capable of self-stressed equilibrium, it is important for each node to have an appropriate number of connections. Yamamoto denotes the maximum number of connections at a node as NC. To determine whether connectivity at each node is sufficient Yamamoto defines the fitness function f_2. When f_2 approaches zero, connectivity at each node is satisfied.

$$f_2 = \sum_{j=1}^{n} \left| \left(NC - \sum_{i=1}^{b} |C(i,j)| \right) \right|$$

6.2.3 Fitness Function 3: *Struts at a node*

For a tensegrity structure to be capable of self-stressed equilibrium, each node should only have one strut. For a *class 1* tensegrity structure to be stable, each node should be connected to exactly one strut (Refer to 2.1 for definition of *class 1*). To determine whether each node has a strut, Yamamoto defines fitness function f_3 as follows [2].

First, Yamamoto computes the number of elements at all nodes denoted CT.

$$CT = diag(\overrightarrow{q_0})^T |C|$$

Yamamoto uses CT to compute the fitness value f_3, which minimizes to zero when each node is connected to exactly one strut.

$$f_3 = \sum_{j=1}^{n} \left| NS - \sum_{i=1}^{b} CT(i,j) \right| \quad where\ NS = NC - 2$$

6.3 Chromosome Encoding of Tensegrity Structures

Chromosome encoding is an important part of using genetic algorithms. Yamamoto proposes a chromosome encoding method for tensegrity structures using the connectivity matrix C and force-density vector $\vec{q}$. Yamamoto's approach uses two chromosomes to represent a tensegrity structure: a chromosome for the connectivity matrix, and a chromosome for the tension coefficient vector.

In the connectivity matrix, each row of the matrix connects two nodes by an edge within the tensegrity structure. Yamamoto labels each row of the matrix as a letter. Yamamoto labels each edge of the tensegrity with a letter. When crossover and mutation are applied, these letters are permuted to create new individuals in the population.

Similar to the connectivity matrix, each row of the tension coefficient vector represents an edge in the tensegrity structure. Yamamoto transforms the tension coefficient vector into a string where each value corresponds to an edge with a related tension coefficient. To generate diversity in the population, the indices of the tension coefficients are permuted through genetic operators.

6.4 Summary of Yamamoto's Method

Yamamoto's method takes an attributed tensegrity structure that is not guaranteed to be capable of self-stressed equilibrium; and returns an attributed tensegrity structure and corresponding node coordinates in a state of self-stress equilibrium. Yamamoto uses genetic algorithms to adjust cable lengths, strut lengths, and connectivity for obtaining a state of self-stressed equilibrium.

Although Yamamoto's work provides inspiration towards the design of our approach explained in Chapter 7, the work presented by Yamamoto has several limitations in our research area. In Yamamoto's method, topology and edge labels (strut or cable) can be changed during the genetic algorithm process for form-finding – only the number of edges remains the same. These changes to the input tensegrity structure would not be feasible in our application (Chapter 7), and would be a considerable limitation to our method. In our proposed method, we assume the tensegrity structure given as input contains correct topology and edge labels, as we adjust edge attributes (resting length, strut length), and node coordinates.

Figure 14 provides a flowchart overview of Yamamoto's method.

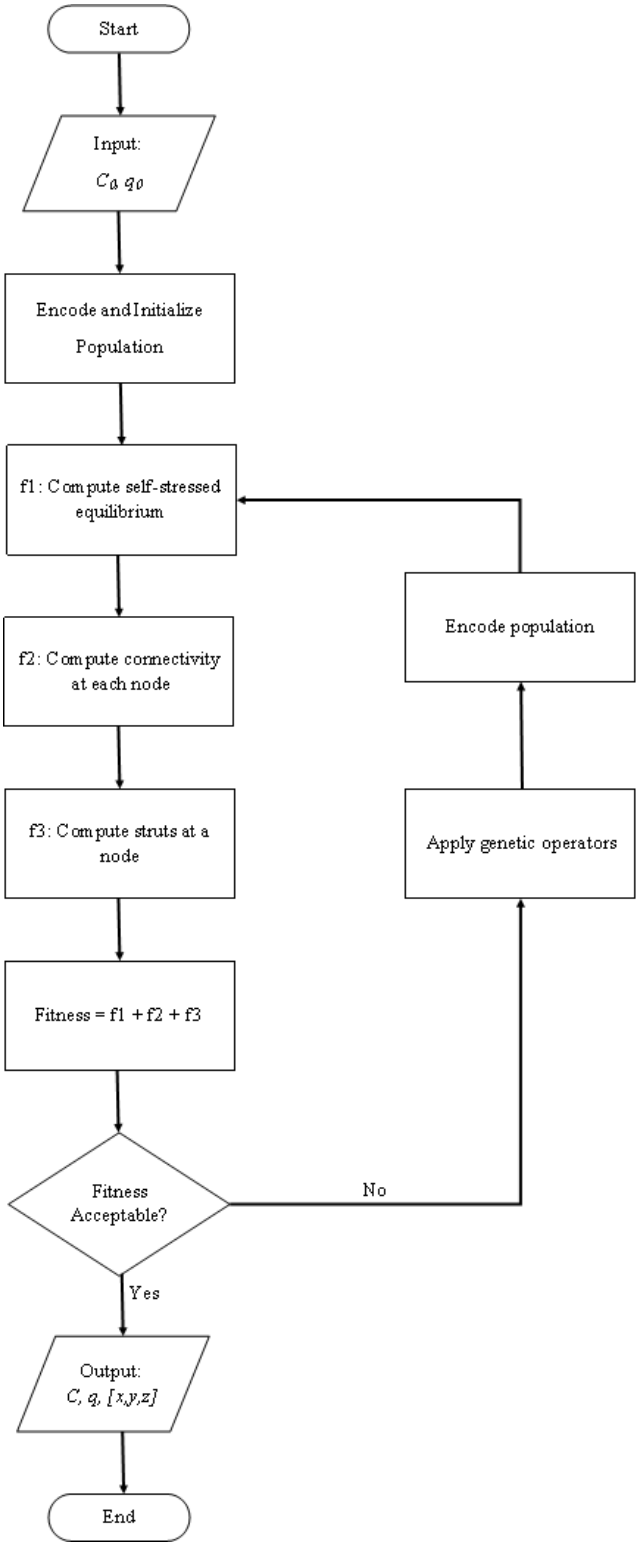

Figure 14 Flowchart illustrating the workflow used by Yamamoto for tensegrity form-finding using genetic algorithms. Figure by Nuwan Perera based on careful investigation of work by Yamamoto et al [2].

Chapter 7

Designing a Genetic Algorithm Approach for Tensegrity Form-Finding

In this chapter we design a stand-alone application for tensegrity form-finding using genetic algorithms. The goal of our application is to reduce iterative adjustments made by researchers when constructing tensegrity structures using simulation support. Our application promotes collaboration and the use of simulation by supporting both *NTRT* and *PushMePullMe3D* platforms. Currently, neither platform supports automated form-finding algorithms, thus requiring iterative adjustments by the user. The iterative adjustment process when constructing tensegrity structures is often time consuming and frustrating for researchers. Although algorithms for tensegrity form-finding applications have been proposed in the literature, implementations are often not available. Therefore, tensegrity researchers cannot capitalize on the benefits of these advances, and rely on experimentation alone.

Our approach to form-finding focuses on reducing the amount of iterative adjustments required to tension, compression, and node coordinates within a structure. For example, when building a full body tensegrity model such as that shown in Figure 6 (Chapter 1), our application will adjust node coordinates to provide the structure with a stable state, or whether a stable state is impossible for the given topology and edge labels. With future extensions, our application will also support user defined fitness functions such as constraints on the spacing of toes within the structure being designed. Using these features, our application aims to be the framework for augmenting the design process for tensegrity structures.

In this chapter we propose an efficient design for a tensegrity form-finding application. Our approach takes inspiration from many proposed algorithms in the field of form-finding. In

our approach, we aim to provide an efficient application that can be executed on an average personal computer (4GB RAM, Intel i5-6300U). The efficiency of the force-density method, allows the application to reach a large audience of users and find stable states of tensegrity structures quickly.

7.1 Application Design Considerations

Several considerations were taken into account during the design of our tensegrity form-finding application. We want our software to be easy to extend, compatible with several simulation platforms, and easy to use. Our application achieves some of these conditions by using MATLAB.

MATLAB is a proprietary programming language used for numerical computing, created by MathWorks. Although MATLAB is an expensive development tool, many institutions hold licenses for the product, making it accessible to most developers in tensegrity research. Using MATLAB allows us to exploit the efficiency of built-in functions for numerical methods. Examples of built-in functions include singular value decomposition and matrix multiplication.

The use of built-in functions allows our code to be short with few files. The compactness of the software makes it easier for other developers to extend or customize our application. To promote collaboration and extensions to our application, sample source code of our fitness function can be found in Appendix A, full source code is available online through a GitHub repository.

By using MATLAB, our application is not constrained to a single simulation platform; rather it can work with data files from any platform as long as a compatible parser is built. If the

simulation platform does not have a parser available, the availability of our code allows other researchers to extend the MATLAB application. By creating a stand-alone application for tensegrity form-finding, we can overcome NDA/IP issues presented by some simulation platforms since the datafiles are usually based on open-source formats even if the simulation source code is proprietary.

Although MATLAB is a proprietary tool to develop on, it can be executed for free by users who do not possess licenses by compiling the MATLAB program. The compiled MATLAB accepts the data file saved from the simulation platform, computes the form-finding for a self-stressed state, and if possible, returns the self-stressed tensegrity structure. Since our MATLAB application will be multi-platform, free to use, and does not require programming knowledge; it should be accessible to most of our users. Figure 14 shows the user interface input of our form-finding application.

Based on our criteria for a stand-alone tensegrity form-finding application, we decided that MATLAB would be a suitable programming language.

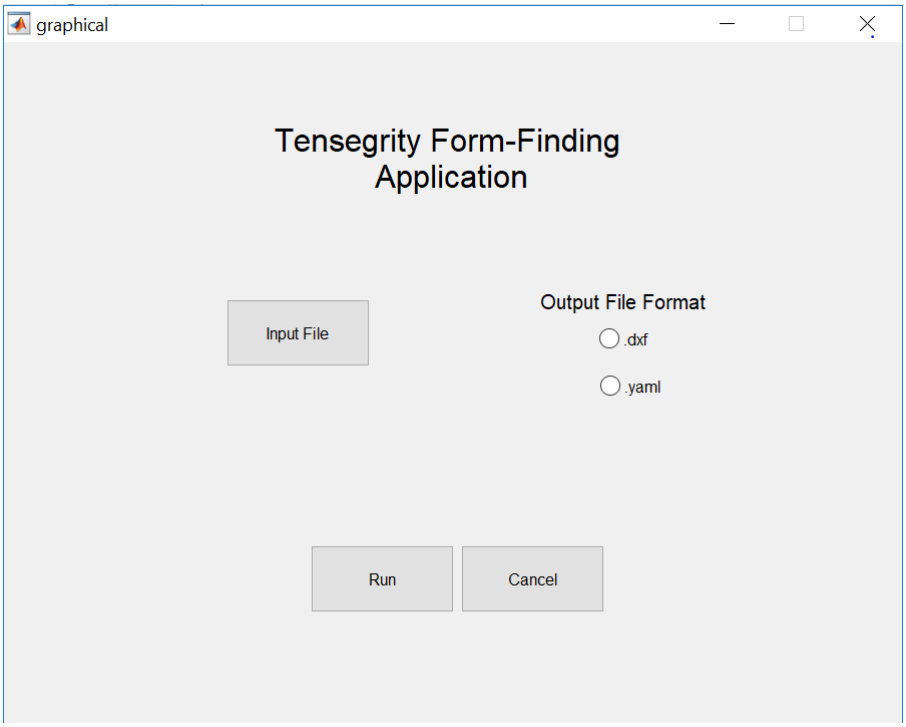

Figure 15 MATLAB input user interface for our tensegrity form-finding application. Figure by Nuwan Perera.

7.2 Design Assumptions for the Prototype Implementation

Since tensegrity form-finding is a very broad area of research, we impose assumptions to constrain our application. The assumptions made by our applications aims to provide sufficient form-finding to as many users as possible.

- **A valid tensegrity structure is given as input.** We assume that the tensegrity structure given to the application as input is valid, according to the criteria we define in Section 2.1.
- **The tensegrity structure is unactuated.** Tensegrity structures can be equipped with actuators that change the length of struts and cables to generate motion. In tensegrity robotics, machine learning can be used to discover actuation patterns [15]. Actuation of tensegrity structures is beyond the scope of this thesis.

- **The connectivity given as input is correct.** We assume that the connectivity given as input to the form-finding application is the desired output.

7.3 Algorithm Inputs

To perform form-finding, the algorithm uses a tensegrity structure created in a simulation platform, either *NTRT* or *PushMePullMe3D*. The structures are accepted into our application as a represented file. For *NTRT,* this can be a *YAML* representation, the markup language used by the platform. For *PushMePullMe3D*, this can be *.dxf*, a common format used to represent CAD models.

The markup files are then parsed to a representation similar to that shown in Fund's work [1]. Each tensegrity structure is represented by a force-density vector $\vec{q}$, edge-node matrix $CONN$, number of nodes n, and x, y, z coordinates of the each node. In other methods, initial node coordinates are unknown [2, 26, 31, 39]. Unlike these methods, we use node coordinate information from the simulation engine as a part of our form-finding input. The use of initial x, y, z coordinates often results in shorter runtime for the form-finding application, as the structure is close to its desired state of self-stressed equilibrium.

Symbol	Description	Type
n	Number of nodes	Integer
$CONN$	Edge-node matrix	$[edge\ \#, node_1, node_2]$ Matrix of size $b \times n$
$coordinates$	Node coordinate matrix	$[x, y, z]$
$\vec{q}_0$	Initial force-density vector	Vector of size $b \times 1$

Table 2 Summary of input parameters for our prototype implementation of the tensegrity form-finding application.

7.4 Design of Tensegrity Form-Finding Approach

Our form-finding approach takes inspiration from Fund's and Yamamoto's algorithms [1, 2]. Unlike Fund's and Yamamoto's methods, our algorithm can make use of initial node coordinates. The node coordinates of each structure are contained within the data file being passed into our application. In our approach, we exploit this additional information to accelerate and improve the form-finding process.

To compute the force-density and equilibrium matrices, we convert the edge-node matrix, $CONN$, into a connectivity matrix C as explained in Section 4.2. The connectivity matrix represents each edge as a row, and each node as a column, where edge k connects node i to j by a connection from 1 to -1 (Figure 10). We choose to use the edge-node matrix as an intermediate storage form because its representation is very similar to both *.yaml* and *.dxf* representations. This makes the form-finding process transparent for both developers and users.

Using the connectivity matrix, we compute the force density matrix D. This approach is analogous to the force-density methods shown by both Fund and Yamamoto [1, 2].

$$D = C^T diag(\overrightarrow{q_0}) C$$

Similarly, we compute the equilibrium matrix A, which determines whether or not the tensegrity structure is in a state of self-stressed equilibrium. Unlike Yamamoto's method, we have access to $\vec{x}, \vec{y}$, and $\vec{z}$ coordinates from the initial design of the tensegrity structure. So we do not need to solve for these coordinates before computing the equilibrium matrix .

$$A = \begin{bmatrix} C^T diag(C\vec{x}) \\ C^T diag(C\vec{y}) \\ C^T diag(C\vec{z}) \end{bmatrix}$$

If the tensegrity structure is in a state of self-stressed equilibrium, the equilibrium matrix multiplied by the force-density vector will be equal to zero [2, 31, 26, 39].

$$A\vec{q} = \vec{0}$$

To determine if the tensegrity structure is acceptable, we solve for the force-density vector $\vec{q}$. Since the equilibrium condition is a homogenous system, multiple solutions for $\vec{q}$ could exist. To solve for $\vec{q}$ we use singular value decomposition (SVD).

$$USV^T = SVD(A)$$

The resulting force-density vector $\vec{q}$ is the last column of the singular matrix V.

The vector $\vec{q}$ represents the force-density at each edge; the magnitude of the force density corresponds to the tension at a given edge. If the value of $\vec{q}$ is positive the corresponding edge is a

tensile element, and the edge is a cable. If the value of $\vec{q}$ is negative, the corresponding edge has no tension, and the edge is a strut.

If the strut and cable configuration of $\vec{q}$ matches the configuration of $\overrightarrow{q_0}$ the tensegrity structure will be self-stressed during simulation. If the strut and cable configurations do not match, the tensegrity structure will not be self-stressed during simulation (Figure 16).

If the tensegrity structure is not acceptable, a genetic algorithm is used to approximate better $\vec{x}, \vec{y}$, and $\vec{z}$ coordinates for the tensegrity structure. Section 7.5 presents the genetic algorithms used for adjusting tensegrity structures to obtain self-stressed equilibrium.

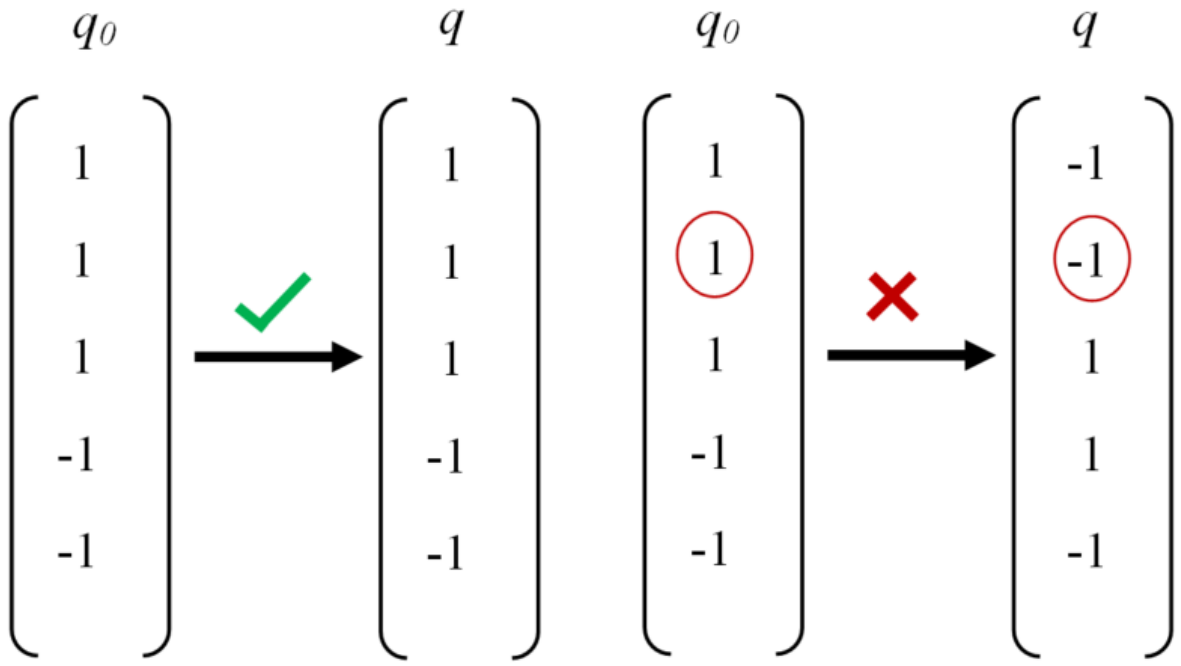

Figure 16 Resulting force-density vector example. Left: shows a tensegrity structure at self-stressed equilibrium, where the configuration of struts are unchanged. Right: shows an extra strut added and changed configuration indicating a failed tensegrity structure. Figure by Nuwan Perera.

7.5 Genetic Algorithms for Structure Adjustment

Genetic algorithms are used to find possible solutions for problems with large search spaces. Tensegrity form-finding presents a vast search space. Even for a simple 3-prism tensegrity structure, there are many possible configurations – although most configurations produce invalid tensegrity structures.

If the tensegrity structure is not self-stressed, the equilibrium equation will not be equal to zero when the new force-density vector $\vec{q}$ is used. We use the sum of the result of the equilibrium equation $\vec{r}$ as the fitness value for evaluating the success of our tensegrity structure.

When a structure is capable of self-stressed equilibrium the $fitness\ value$ should minimize to zero.

$$A\vec{q} = \vec{r}$$

$$fitness\ value = SUM(abs(\vec{r}))$$

Based on the success criteria, the genetic algorithm will create a population using crossover and mutation. The fittest individuals in the population will be selected to create the resulting tensegrity structure.

For efficient computation of genetic algorithms, we use the MATLAB Optimization Toolbox. The Optimization Toolbox is a built-in facility MATLAB that allows users to define fitness function and execute genetic algorithms. Although the MATLAB toolbox is used, the chromosome encodings, representations, and operations closely resemble those in Chapter 5. Figure 17 also provides further high-level details about the genetic algorithm used in our method. The use of the toolbox improves the efficiency of our genetic algorithm implementation compared to a manual implementation.

7.6 Output of Form-Finding Application

Once a feasible tensegrity structure is found, it is returned to the user using the same representation as the input given. For example, if a *.dxf* file was given as input, the user will receive a *.dxf* file returned as output. The new file will contain a tensegrity structure that is capable of self-stressed equilibrium. It is possible that parameters within the simulation platform such as damping and prestress could need adjustment for the structure to be stable. Simulation

parameter information is not obtainable through current data representations so these factors cannot be accounted for in the form-finding process.

Figure 17 shows a flowchart of our form-finding method similar to the flowchart of Yamamoto's method shown in Figure 14.

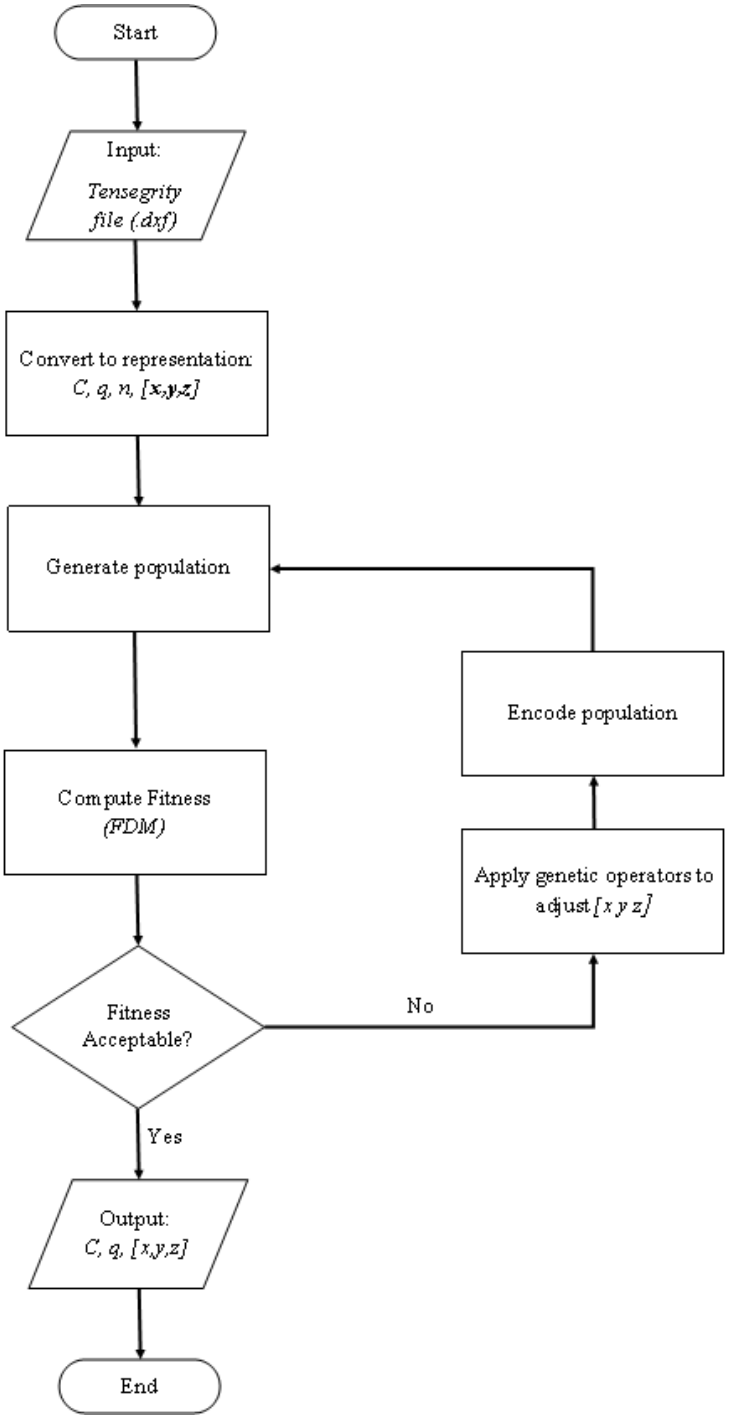

Figure 17 Figure of our tensegrity form-finding application presented in Chapter 7.

Chapter 8

Results and Analysis of Tensegrity Form-Finding

In this chapter, we present the results and prototype implementation of our tensegrity form-finding application. We also provide an analysis of our findings and limitations in our current implementation of the form-finding application.

8.1 Limitations of Our Prototype Implementation of Tensegrity Form-Finding

In this thesis, we implement the fitness function and genetic algorithm component of our designed form-finding application from Chapter 7.

Developing a robust tensegrity form-finding application is a challenging task. Several months were spent reviewing the literature for published form-finding approaches, and implementing these proposed approaches [1, 2, 25, 31] . Fund's thesis work provided clear explanations of form-finding for shell structures as well as well described MATLAB code for testing both the force-density method and Pucher's method for shell structures. Aside from Fund's thesis, other publications do not make their source code publicly available [1]. However, Fund's approach is for the form-finding of shell structures, and requires considerable modifications to work for tensegrity structures. For example, both publications by Yamamoto and Gan do not have source code publicly available [2, 31]. During this thesis we attempted to acquire source code for both methods through communication with the authors; however, this was not made available. The absence of source code made the reimplementation of Yamamoto's and Gan's tensegrity form-finding approaches a challenging and time consuming task [2, 31]. After careful implementation, we experienced several ambiguities re-implementing the works of both Yamamoto and Gan [2, 31].

Our design of a tensegrity form-finding application outlined in Chapter 7 takes inspiration from several existing form-finding methods. During implementation, several months were spent understanding the numerical methods involved in tensegrity form-finding. Our prototype implementation is able to successfully conduct form-finding for 2-dimensional tensegrity structures. Even after months of investigation, the prototype implementation has instabilities when applied to 3-dimensional structures. The limitations of our implementation led to unexpectedly poor results for some test cases. Two limitations that affected the functionality of our program are: the free variables present when fixed nodes are eliminated, and the homogenous properties of the equilibrium equation.

Although our prototype has limitations in functionality, it still provides a good foundation for future work in tensegrity simulation and form-finding research. Proof of concept is provided by successfully conducting form-finding on 2-dimensional tensegrity structures. Instabilities for 3-dimensional structures can be attributed to the equations and numerical instabilities faced in 3-dimensions.

Unlike membrane shell structures, tensegrity structures do not have fixed nodes [39]. This creates several free variables in the force-density method. The free variables lead to many possible solutions to the system of equations because they make the force-density equation a homogenous system.

$$D\ [\vec{x}\ \vec{y}\ \vec{z}] = \vec{0}$$

Some researchers including Lee et al use genetic algorithms to find the best solution to the force-density equation [33].

Similar to the force-density equation, the self-stressed equilibrium equation is also a homogenous system. The homogenous system means that the equation can have multiple solutions. For the force-density method, this means we can obtain multiple solutions to the force-density equation. Our MATLAB implementation and design method returns the smallest non-zero solution. This is often not the best solution to be used for form-finding of tensegrity structures; for example, it might be a rigid non-tensegral structure where most of the edges are struts.

Ongoing attempts are being made to overcome the drawbacks of the homogenous systems, but so far none have resolved the stability problems for 3-dimensional cases. The best results were produced by taking the smallest non-zero solution by using singular value decomposition.

Other researchers have also noted challenges with tensegrity form-finding in areas of numerical stability and convergence. Most current tensegrity form-finding approaches limit the algorithms to very specific cases of tensegrity structures: regular or irregular tensegrity structures, symmetric tensegrity structures, multiple states of self-stress [31, 33, 37].

Significant effort and time was spent implementing an efficient fitness function, thus limiting the time available to address the numerical instability problems and produce a fully robust form-finding application within the scope of this thesis.

8.2 Validating the Form-Finding Algorithm

To illustrate the strengths and limitations of our prototype implementation, we use four test cases: a 2-dimensional tensegrity 'X', a self-stressed 4-strut tensegrity structure, a self-stressed 3-strut tensegrity prism, and an unstable 3-strut tensegrity prism. The 2-dimensional tensegrity 'X' allows us to validate the correctness of the algorithm in two dimensions. The 4-strut tensegrity structure is chosen for testing to compare against other methods such as the form-finding method proposed by Gan [31] . The 3-strut prism is chosen as a test case because of its simplicity; it is the most basic form of a 3-dimensional tensegrity structure, thus making a good test case to verify the correctness of our algorithm.

In this section, we present the results of our fitness function, to determine whether or not a tensegrity structure is currently capable of self-stressed equilibrium. The results from this section illustrate the strengths and limitations of our current implementation of the form-finding application. Section 8.3 provides an analysis of the results presented in this section.

8.2.1 Results: 2-strut Tensegrity 'X'

To validate the correctness of our approach, we use a 2-strut tensegrity 'X', a 2-dimensional tensegrity structure similar to that used by Moored and Bart-Smith [40]. The correct output of a 2-dimensional tensegrity 'X' can be computed manually for verification. Table 3, 4, and 5 show the input parameters for a 2-dimensional tensegrity 'X', which satisfies the condition $A\vec{q} = \vec{0}$.

Number of Nodes (n) = 4

x	y
0	1
0	0
1	0
1	1

Table 3 Input *x* and *y* parameters for a 2-strut tensegrity 'X' in 2-dimensional space.

q_0
1
1
1
1
-1
-1

Table 4 Input force-density vector for a tensegrity 'X' in 2-dimensional space.

Edge #	Node 1	Node 2
1	1	2
2	2	3
3	3	4
4	1	4
5	1	3
6	2	4

Table 5 Input connectivity of a 2-strut tensegrity 'X'.

8.2.2 Results: 4-strut Tensegrity Prism

The 4-strut tensegrity prism used in the tensegrity form-finding approach by Gan et al is used to validate our proposed form finding application [31]. Tables 6, 7, and 8 show the input parameters. Table 9 shows the output force-density vector returned by the fitness function.

Number of Nodes (n) = 8

x	***y***	***z***
-0.04699	-0.61277	-0.39280
-0.10034	-0.26753	0.36964
0.46218	-0.15565	0.36840
0.50295	0.101945	-0.18404
-0.63877	-0.23305	-0.02637
-0.33470	0.33128	0.48001
-0.29177	0.47038	-0.06329
-0.01584	0.36591	-0.55163

Table 6 Input *x, y, z* parameters for tensegrity form-finding based on a stable 4-strut tensegrity structure from Gan et al.

q_0
1.5331
3.2160
4.9010
1.6192
2.8350
1.8900
1.7700
1.3200
3.4200
2.7530
1.6500
5.0500
-3.1600
-2.0000
-3.7120
-3.2504

Table 7 Input force-density vector for tensegrity form-finding based on a stable 4-strut tensegrity structure from Gan et al.

Edge #	Node 1	Node 2
1	1	2
2	2	3
3	3	4
4	1	4
5	5	6
6	6	7
7	7	8
8	5	8
9	1	5
10	2	6
11	3	7
12	4	8
13	1	6
14	2	7
15	3	8
16	4	5

Table 8 Edge-node matrix based on a stable 4-strut tensegrity structure by Gan et al.

q
0.1498
0.2113
0.4138
0.1342
0.2272
0.1908
0.1169
0.0978
0.2938
0.2858
0.0994
0.4302
-0.2824
-0.1815
-0.2900
-0.2689

Table 9 Output force-density vector result from tensegrity form-finding application for 4-strut tensegrity structure.

8.2.3 Results: Stable 3-strut Tensegrity Prism

The stable 3-strut tensegrity prism is created in *PushMePullMe3D* (Figure 18). The data from the *.dxf* file is used as input data for our tensegrity form-finding application. Tables 10, 11, and 12 show the input parameters. Table 13 shows the output force-density vector returned by the fitness function.

Number of Nodes (n) = 6

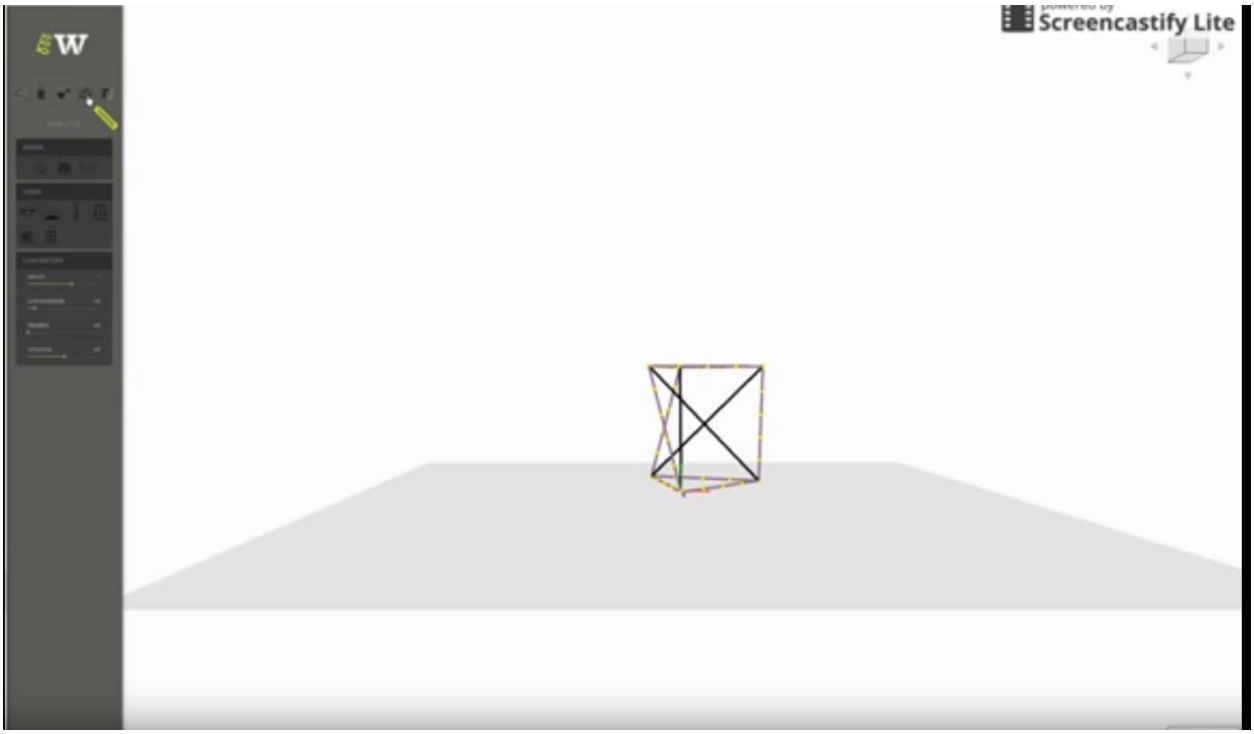

Figure 18 Stable tensegrity 3-prism created in PushMePullMe3D.

x	*y*	*z*
159.42	138.82	0.00
-56.91	31.85	220.33
-45.79	206.23	0.00
155.17	72.02	219.99
-1.56	5.19	0.00
14.35	235.56	220.39

Table 10 Input *x, y, z* parameters for tensegrity form-finding based on a stable 3-strut tensegrity prism.

q_0
-1
-1
-1
1
1
1
1
1
1
1
1

Table 11 Input force-density vector for tensegrity form-finding based on a stable 3-strut tensegrity prism.

Edge #	Node 1	Node 2
1	1	2
2	3	4
3	5	6
4	1	3
5	1	5
6	1	6
7	2	3
8	2	4
9	2	5
10	3	6
11	4	6
12	4	5

Table 12 Edge-node matrix based on a stable 3-strut tensegrity prism.

q
0.3032
-0.3030
0.0002
0.2496
-0.3604
-0.3482
0.0002
-0.2503
-0.3065
0.3068
0.3612
0.34848

Table 13 Output force-density vector result from tensegrity form-finding application.

8.2.4 Results: Unstable 3-prism

To validate that our application can identify unstable tensegrity structures, we create a tensegrity 3-prism that collapses when simulation is applied in *PushMePullMe3D* (Figure 19). The data from the *.dxf* file is used as input data for our tensegrity form-finding application. Tables 14, 15, and 16 show the input parameters. Table 17 shows the output force-density vector returned by the fitness function.

Number of Nodes (n) = 6

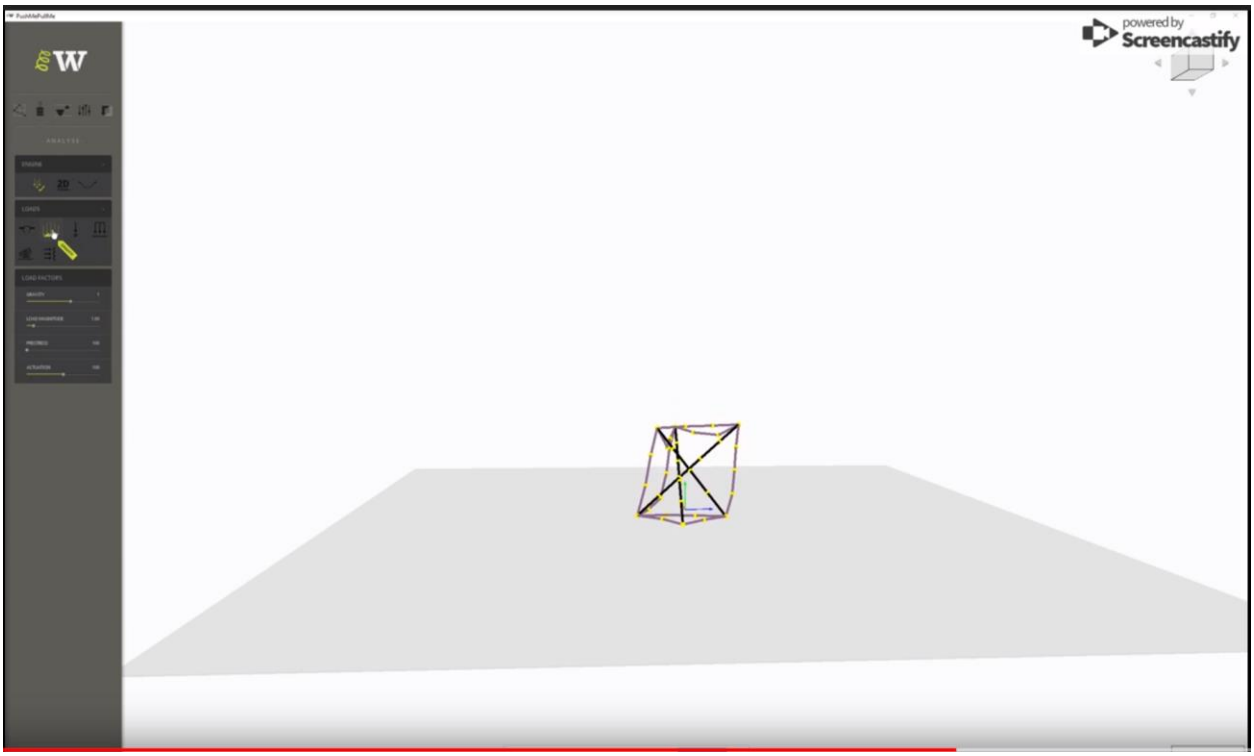

Figure 19 An unstable 3-prism tensegrity structure collapsing in PushMePullMe3D. The buckling of the cables (purple) is shown as the structure collapses.

x	*y*	*z*
-100.00	0.00	0.00
100.00	0.00	200.00
-29.53	114.73	-3.50
-29.53	85.27	196.50
8.89	99.98	13.67
8.89	100.02	213.67

Table 14 Input *x, y, z* parameters for tensegrity form-finding based on an unstable 3-strut tensegrity prism.

q_0
-1
-1
-1
1
1
1
1
1
1
1
1

Table 15 Input force-density vector for tensegrity form-finding based on an unstable 3-strut tensegrity prism.

Edge #	Node 1	Node 2
1	1	2
2	3	4
3	5	6
4	1	3
5	1	5
6	1	6
7	2	3
8	2	4
9	2	5
10	3	6
11	4	6
12	4	5

Table 16 Edge-node matrix based on an unstable 3-strut tensegrity prism.

q
-0.5107
-0.0076
-0.2536
-0.0606
0.3840
0.4937
0.2645
0.1589
0.3072
-0.2890
-0.0769
-0.0244

Table 17 Output force-density vector result from tensegrity form-finding application.

8.3 Analysis of Results

This section presents the analysis of our results from Section 8.2.

8.3.1 Stable 2-Strut Tensegrity 'X'

For the 2-strut tensegrity 'X', our equation satisfies the equilibrium condition of $A\vec{q} = \vec{0}$. In general, our algorithm is able to robustly conduct 2- tensegrity form-finding for 2-dimensional problems.

8.3.2 Stable 4-Strut Tensegrity Structure

Our form-finding application showed that the 4-strut tensegrity structure presents expected results as the form-finding fitness function identified the correct placement of cables and struts. When simulated, this structure resulted in a self-stressed tensegrity structure, meaning that our form-finding application was successful. However, the force-density values returned by the form-finding application is not consistent with the known output values published by Gan et al [31]. The values are different because multiple solutions exist to the equilibrium equation, and our approach does not always find the best solution to the equilibrium equation. When a close to zero solution is found, our approach gets stuck in a local minimum; a limitation of our current algorithms numerical stability.

8.3.3 Stable 3-Strut Tensegrity Prism

For the stable 3-strut tensegrity prism we designed in *PushMePullMe3D*, the tensegrity structure was not determined to be stable by our form-finding application. The resulting force-density vector increases the number of struts required from 3 struts (truth) to 5 struts. Although this would create a stable structure, this is not a tensegrity structure. This means that our 3-strut prism test case fails. Section 8.3 presents possible explanations for this failure.

8.3.4 Unstable 3-Strut Tensegrity Prism

For the unstable 3-strut tensegrity prism we designed in *PushMePullMe3D*, the tensegrity structure was not determined to be stable by our form-finding application. The resulting force-density vector increases the number of struts required from 3 struts (truth) to 8 struts. This is a possible solution producing a stable structure, however, this result is not a valid tensegrity structure. Increased iterations and fine adjustment of x, y, z coordinates could result in a valid tensegrity structure.

8.4 Limitations of our Form-Finding Application

Based on our results presented in this chapter, limitations of our implementation are apparent. Many of our 3-dimensional cases presented in this chapter did not produce successful results. This is attributed to numerical instabilities present in our implementation. The equilibrium equation we present in Chapter 7 has multiple solutions since it is a homogeneous system of equations. Finding a suitable non-zero solution to this problem presented instabilities and challenges for our implementation. In our approach, we use singular value decomposition (SVD) to find the smallest non-zero solution. In some cases, SVD can return expected results (Section 8.2.2), however, in other cases, the results can be unexpected (Section 8.2.3). Therefore, our solution presented is not entirely robust.

In an attempt to improve robustness of our approach, we consulted with Dr. Keith Moored from Lehigh University at the Conference on Biopropulsion of Adaptive Systems, Queen's University Biological Station, July 23-26, 2018. Dr. Moored's PhD work involved mechanical engineering and tensegrity structures [40]. From these conversations, we were not able to improve the robustness of our approach; however, his suggestion to test the 2-dimensional case of the tensegrity 'X' (Section 8.2.1), showed correctness of our approach.

To continue improving the robustness of our approach and overcome numerical instabilities, expertise in mechanical engineering is required. If added expertise solves our numerical instabilities, we provide a useful contribution to future work in tensegrity form-finding and simulation research (Section 9.2).

Chapter 9

Conclusion

This thesis surveys the state of the art in tensegrity form-finding, and presents the design and implementation of a tensegrity form-finding application using genetic algorithms. The underlying goal of this thesis is to improve the usability of tensegrity simulation software by augmenting the design process used for tensegrity structures. Our form-finding application aims to augment the design process for tensegrity researchers by reducing the number of iterative adjustments required to obtain for a tensegrity structure that satisfies the designer.

9.1 Summary of Contributions

This thesis makes several contributions to the field of tensegrity research, and its intersection with computer science. Here we reiterate the contributions presented in Section 1.1.

9.1.1 Identify the need for form-finding of tensegrity structures

Tensegrity structures are highly intricate and difficult to construct. Many researchers can spend weeks or months constructing physical tensegrity structures. Once these structures are constructed, they still might not be considered successful for the experiment. Computer simulation shows promise for advancing tensegrity research; reducing the cost of construction, providing better results, increasing access and promoting collaboration.

The intricacies of tensegrity construction creates a challenge for many researchers in both physical construction and computer simulation. Based on several conversations with tensegrity researchers including Tom Flemons and Dr. Richard Gordon on the challenges of tensegrity

research, we determined that a computational approach should be taken for obtaining a state of self-stressed equilibrium. Often researchers will design tensegrity structures for an experiment, only for the structure to collapse during simulation. Tensegrity structures collapse when they are not in a state of self-stressed equilibrium. For some researchers, obtaining a state of self-stressed equilibrium can take weeks or months of iterative adjustments. Computational approaches for iterative adjustment through form-finding shows promise in reducing the amount of iterative adjustments required by the researcher when designing a tensegrity structure. The computational approach taken in this thesis is the development of a machine learning based tensegrity form-finding application (Chapter 7).

9.1.2 Identify Simulation Support for Augmenting Tensegrity Design

This thesis contributes to the research of Dr. Blostein's research group at Queen's University by identifying a candidate tensegrity simulation platform for future research.

Several simulation platforms are capable of simulating tensegrity structures. In this thesis, we identify the simulation platform suitable for the research of our research group and collaborators: Dr. Dorothea Blostein, Dr. Richard Gordon, and Tom Flemons. In selecting a platform for simulation, we extensively surveyed both *NTRT* and *PushMePullMe3D*. In this thesis, we choose *PushMePullMe3D* because of its monolithic software architecture, intuitive user interface, and ability to support extensions.

Unlike *NTRT* which is open-source, the source-code for *PushMePullMe3D* is proprietary, requiring legal agreements for development. Through this thesis, we continue collaboration with Dr. Gennaro Senatore, the creator of *PushMePullMe3D*. Establishing collaboration with Dr.

Senatore has been a considerable contribution to our research group, bringing in structural and mechanical engineering expertise.

Selecting *PushMePullMe3D* as the primary simulation platform for our research group has created a foundation for future tensegrity research. By improving the quality of *PushMePullMe3D* we also promote collaboration and accessibility for tensegrity research and simulation.

9.1.3 Survey State-of-the-Art Form-Finding Algorithms

Form-finding is an approach for finding states of equilibrium for a tensegrity structure. In this thesis, we review multiple form-finding approaches. Form-finding was first proposed by Schek in 1974 for membrane shell structures. In Chapter 3, we summarize two form-finding methods used for membrane shell structures: Pucher's method and the force-density method [1]. Although Schek's force-density method is designed for membrane shell structures, a modified method is used for tensegrity form-finding.

With several free variables present, tensegrity structures are less constrained than membrane shell structures (no free nodes, all forces are internal). Researchers have taken different approaches to overcome these free variables for tensegrity form-finding. Two common approaches taken are numerical and analytical approaches, and the use of machine learning. For this thesis, considerable effort was spent understanding the mathematics and mechanics of many tensegrity form-finding algorithms. In Chapter 6, we outline a machine learning approach for tensegrity research by Yamamoto et al.

9.1.4 Design and Prototype Implementation of a Tensegrity Form-Finding Application

To address the challenges of tensegrity structure design, we design a form-finding application for tensegrity structures (Chapters 7 and 8). The design of our application is based on the state of the art in tensegrity form-finding research, including inspiration from Schek's force-density method and modern machine learning approaches to form-finding.

The goal of our form-finding application is to augment the design process for tensegrity structures in computer simulation. We use form-finding to reduce the amount of fine iterative adjustments required by researchers to obtain a self-stressed tensegrity structure that will not collapse when simulated. The application designed in this thesis also aims to be the first automated form-finding application for tensegrity simulation, compatible with both *NTRT* and *PushMePullMe3D*. Currently, both platforms conduct form-finding mainly through manual experimentation. The introduction of an automated form-finding algorithm can improve the user experience and quality of simulations for tensegrity researchers.

Although the form-finding application in this thesis has limited functionality, we lay a foundation for future research in tensegrity form-finding and computer simulation. Our application shows promise to improve the design process for tensegrity researchers, improving the user experience for researchers who would otherwise be deterred from computer simulation.

9.2 Future Work

The design and implementation of a tensegrity form-finding application, requires expertise in several areas including computer science, mechanical engineering, structural engineering, and a knowledge of tensegrity structures. This thesis lays the groundwork for future extensions, improvements, and collaboration for tensegrity research.

The form-finding approach proposed in this thesis has limited functionality for 3-dimensional cases, so as it stands the prototype would produce inadequate results for end users. Although the results are limited in this application, we set the groundwork for future work in building in a stand-alone tensegrity form-finding application. With more mathematics and mechanical engineering expertise during implementation, our prototype can be expanded to conduct form-finding on 3-dimensional tensegrity structures.

Once the form-finding algorithms in the prototype have been made robust, the application can be extended to allow users to extend new parsers and define their own fitness functions for tensegrity structures. These additional features would provide added benefit for researchers who have intermediate to advanced programming skills looking for increased functionality from our form-finding application. Particularly good extensions would be the addition of compatibility with CAD design software such as *SketchUp* and *Rhino 3D*. These two platforms are commonly used to design tensegrity structures within our research group as well as by the broader research community.

Further research can also explore the search space of tensegrity form-finding using other searching and machine learning methods such as simulated annealing or Monte Carlo simulation.

Our form-finding application, provides a good starting point for promoting the use of computer simulation in tensegrity research as we aim to reduce the fine iterative adjustments required for this process.

References

[1] A. I. Fund, "Form-Finding Structures," MIT, Cambridge, 2008.

[2] M. Yamamoto, B. S. Gan, K. Fujita and J. Kurokawa, "A genetic algorithm based form-finding for tensegrity structure," in *The Twelfth East Asia-Pacific Conference on Structural Engineering and Construction*, Hong Kong, 2011.

[3] B. Burkhardt, "Tensegrity 3-prism," 2004. [Online]. Available: https://commons.wikimedia.org/wiki/File:Tensegrity_3-Prism.png. [Accessed 9 April 2018].

[4] V. G. Jauregui, Tensegrity and their Application to Architecture, Servicio de Publicaciones Universidad de Cantabria, 2010.

[5] T. Flemons and D. Blostein, "New Approaches to Mechanizing Tensegrity Structures," in *ASCE Earth and Space Conference*, Cleveland, 2018.

[6] R. E. Skelton and M. C. de Oliveira, Tensegrity Systems, New York: Springer, 2009.

[7] H. J. Schek, "The force density method for form finding and computation of general networks," *Computer methods in applied mechanics and engineering,* vol. 3, no. 1, pp. 115-135, 1974.

[8] B. FrantzDale, "3-Tensegrity," Wikimedia Commons, 7 December 2008. [Online]. Available: https://en.wikipedia.org/wiki/File:3-tensegrity.svg. [Accessed 30 July 2018].

[9] R. Motro, "Tensegrity: from Art to Structural Engineering," in *2012 IASS-APCS Symposium*, Seoul, 2012.

[10] R. L. Swanson, "Biotensegrity: a unifying theory of biological architecture with applications to osteopathic practice, education, and research - a review and analysis," *Journal of the American Osteopathic Association,* vol. 113, no. 1, pp. 34-52, 2013.

[11] T. Flemons, "The Geometry of Anatomy," *(Self published) from http://www.intensiondesigns.com/geometry_of_anatomy.html,* 2007.

[12] T. Flemons, "Bones of Tensegrity," *(Self published) from http://www.intensiondesigns.com/bones_of_tensegrity.html,* 2012.

[13] N. Gordon and R. Gordon, Embryogenesis Explained, Singapore: World Scientific, 2016.

[14] D. Ingber, "Cellular tensegrity: defining new rules for biological design that govern the cytoskeleton," *Scientific American,* vol. 52, pp. 48-57, 1998.

[15] K. Caluwaerts, J. Despraz, A. Iscen, A. P. Sabelhaus, J. Bruce, B. Schrauwen and V. SunSpiral, "Design and control of compliant tensegrity robots through simulation and hardware validation," *Journal of the Royal Society Interface,* vol. 11, no. 98, 2014.

[16] NASA Tensegrity Robotics Toolkit, "NTRT Learning Library," 2015. [Online]. Available: http://ntrtsim.readthedocs.io/en/latest/learning-library-walkthrough.html#algorithm-options. [Accessed 20 10 2016].

[17] S. H. Juan and J. M. Miratz Tur, "Tensegrity frameworks: Static analysis review," *Mechanism and Machine Theory,* vol. 43, pp. 859-881, 2008.

[18] BAR Photography, "Needle Tower," Tensegrity Wiki, 30 July 2018. [Online]. Available: http://tensegritywiki.com/Needle+Tower. [Accessed 30 July 2018].

[19] M. Donald, "Kurilpa Bridge," Wikimedia Commons, 28 July 2009. [Online]. Available: https://commons.wikimedia.org/wiki/File:KP_Kurilpa_Bridge_IMG_0717_(3768746960).jpg. [Accessed 30 July 2018].

[20] Sunspiral, "NASA Superball Tensegrity Lander Prototype," Wikimedia Commons, 1 October 2014. [Online]. Available: https://commons.wikimedia.org/wiki/File:NASA_SUPERball_Tensegrity_Lander_Prototype.jpg. [Accessed 30 July 2018].

[21] R. W. Burkhardt, A Practical Guide to Tensegrity Design, Cambridge: Cambridge University Press, 2005.

[22] R. E. Skelton, R. Adhikari, J. Pinaud and W. Chan, "An Introduction to the Mechanics of Tensegrity Structures," in *40th IEEE Conference on Decision and Control*, Orlando, 2001.

[23] W.-F. Chen and E. M. Lui, Handbook of Structure Engineering, Second Edition, Boca Raton: CRC Press, 2005.

[24] J. Connor, S. Lamar and J. P. Woolf, "Automatic Solution of Pucher's Equation," *Journal of the Structural Division,* vol. 93, no. 2, pp. 359-378, 1967.

[25] G. G. Estrada, H. J. Bungartz and C. Mohrdieck, "Numerical form-finding of tensegrity structures," *Journal of Solids and Structures,* vol. 43, no. 22, pp. 6855-686, 2006.

[26] K. Koohestani, "Form-finding of tensegrity structures via genetic algorithms," *Journal of Solids and Structures,* vol. 49, no. 5, pp. 739-747, 2012.

[27] K. Koohestani and S. D. Guest, "A new approach to the analytical and numerical form-finding of tensegrity structures," *International Journal of Solids and Structures,* vol. 50, pp. 2995-3007, 2013.

[28] Y. Li, X.-Q. Feng, Y.-P. Cao and H. Gao, "A Monte Carlo form-finding method for large scale regular and irregular tensegrity structures," *International Journal of Solids and Structures,* vol. 47, pp. 1888-1898, 2010.

[29] C. Paul, H. Lipson and F. J. V. Cuevas, "Evolutionary form-finding of tensegrity structures," in *ACM Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, Washington, 2005.

[30] X. Xu and Y. Luo, "Form-finding of nonregular tensegrities using a genetic algorithm," *Mechanics Research Communications,* vol. 37, pp. 85-91, 2010.

[31] B. S. Gan, J. Zhang, D.-K. Nguyen and E. Nouchi, "Node-based genetic form-finding of irregular tensegrity structures," *Computers and Structures,* vol. 159, pp. 61-73, 2015.

[32] Y. Chen, J. Feng and Y. Wu, "Novel form-finding of tensegrity structures using ant colony systems," *ASME Journal of Mechanical Robotics,* vol. 4, no. 3, p. Article No 031001, 2012.

[33] S. Lee, J. Lee and J. Kang, "A Genetic Algorithm Based Form-finding of Tensegrity Structures with Multiple Self-stress States," *Journal of Asian Architecture and Building Engineering,* vol. 16, no. 1, pp. 155-162, 2017.

[34] M. Mitchell, An introduction to genetic algorithms, Cambridge: MIT Press, 1996.

[35] D. E. Goldberg and J. H. Holland, "Genetic Algorithms and Machine Learning," *Machine Learning,* vol. 3, no. 2-3, pp. 95-99, 1988.

[36] R. Motro, Tensegrity: Structural Systems for the Future, London: Elsevier, 2003.

[37] R. Connelly and M. Terrell, "Globally rigid symmetric tensegrities," *Structural Topology,* vol. 21, pp. 59-78, 1995.

[38] R. Connelly, "Rigidity and Energy," *Inventiones mathematicae,* vol. 66, pp. 11-33, 1982.

[39] A. Harichandran and I. Y. Sreevalli, "Form-Finding of Tensegrity Structures based on Force Density Method," *Indian Journal of Science and Technology,* vol. 9, no. 24, pp. 1-6, 2016.

[40] K. W. Moored and H. Bart-Smith, "Investigation of clustered actuation in tensegrity structures," *International Journal of Solids and Structures,* vol. 46, pp. 3272-3281, 2009.

[41] C. Paul, H. Lipson and F. J. V. Cuevas, "Evolutionary Form-Finding of Tensegrity Structures," in *GECCO*, Washington, 2005.

Appendix A

This appendix contains a copy of the source code for the 3-dimensional case fitness function described in Chapter 7.

```
function [ fitness ] = tensegrity_FDM_fitness(n, CONN, q, coordinates )
%TENSEGRITY_FDM_FITNESS This function determines whether or not a
%tensegrity structure is self-stressed, thus at self-stressed equilibrium.
%   We use to force density method (Schek (1974), Fund (2008)) to determine
%   whether or not a tensegrity structure is at self-stressed equilibrium
%
% Input variables:
% C - Connectivity matrix
% n - number of nodes
% q - force-density vector
% coordinates - x,y,z nodal coordinates of tensegrity structure
%
%
% Output variable:
% fitness - determines the fitness of the structure
% The output should minimize to zero for a self-stressed structure
[m,~] = size(CONN);
C = zeros(m,n);
% Populate Connectivity Matrix C
for j = 1:m
    r = CONN(j,2);
    C(j,r) = 1;
    s = CONN(j,3);
    C(j,s) = -1;
end

x = coordinates(:,1);
y = coordinates(:,2);
z = coordinates(:,3);

% Compute self-equilibrium matrix
A = [C'*diag(C*x); C'*diag(C*y); C'*diag(C*z);];

% Solve for Aq = 0 using SVD
[~,~,V] = svd(A);
new = V(:, end);

fitness = sum(abs(A*new));

end
```