Universidade Federal de Pernambuco
Centro de Informática

Graduação em Engenharia da Computação

TRUSS BRIDGES CREATION AND STRUCTURAL MASS OPTIMIZATION WITH EVOLUTIONARY ALGORITHM

Hugo Lispector

TRABALHO DE GRADUAÇÃO

Recife
16 de maio de 2022

Universidade Federal de Pernambuco
Centro de Informática

Hugo Lispector

TRUSS BRIDGES CREATION AND STRUCTURAL MASS OPTIMIZATION WITH EVOLUTIONARY ALGORITHM

Trabalho apresentado ao Programa de Graduação em Engenharia da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação.

Orientador: *Paulo Salgado Gomes de Mattos Neto*

Recife
16 de maio de 2022

To my parents, who educated me with love and by example.

To Marco, for being an ingenious and easygoing friend during all my university years.

To Professor Odilon, for treating us students with trust and kindness.

כל העולם כולו גשר צר מאוד והעיקר לא לפחד כלל

The whole world is a very narrow bridge and the most important thing is not to fear at all.

— NACHMAN OF BRESLOV

ABSTRACT

Trusses are structures that consist of straight members connected by joints located at their endpoints. This type of structure is one of the most used for engineering applications and studies regarding computational optimization of trusses have become very popular recently, with hundreds of publications every year.

One of the main uses of trusses are bridges. In this project, an application that is able to generate and optimize planar truss bridges using an evolutionary algorithm was developed. This research focuses on the minimization of structural mass through a search of design variable concerning topology and shape. An intuitive and friendly graphical user interface interface was also created allowing users to visually set parameters of the desired bridge to be optimized.

After multiple experiments, it was possible to demonstrate that the developed algorithm was converging into feasible solutions. Moreover, the developed system has the potential to give insights and help architects and engineerings in their truss bridges design projects.

Keywords: Evolutionary Algorithm, Truss Structure, Truss Bridges, Truss Optimization, Structural Mass Optimization, Topology and Shape Optimization.

RESUMO

Treliças são estruturas formadas por membros retilíneos conectados através de nós em suas extremidades. Esse tipo de estrutura é uma das mais utilizadas em aplicações de engenharia e estudos sobre a otimização de treliças se tornaram bastante populares recentemente, com centenas de publicações anualmente.

Uma das principais aplicações de treliças são pontes. Neste projeto, foi desenvolvido um aplicativo capaz de gerar e otimizar pontes de treliças através do uso de um algoritmo evolutivo. Esta pesquisa foca na minimização de massa estrutural através da busca de variáveis de design relacionadas a topologia e forma. Uma interface gráfica intuitiva e amigável também foi desenvolvida. Tal recurso permite aos usuários a configuração dos parâmetros das pontes a serem otimizadas de forma visual.

Após vários experiments, foi possível demonstrar que o algoritmo desenvolvido estava convergindo e gerando soluções viáveis. Além disso, o sistema criado tem o potencial de fornecer ideias e ajudar engenheiros e arquitetos na criação de designs e projetos de pontes de treliças.

Palavras-chave: Algoritmo Evolutivo, Estrutura de Treliças, Pontes de Treliças, Otimização de Treliças, Minimização de Massa em Estruturas, Otimização de Topologia e Forma.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

Trusses are one of the most used structures in engineering applications for its simplicity and low cost of construction [22]. This type of structure is composed of slender members connected together at their extremities [13]. When creating truss bridges, engineers and architects will broadly be given at least two fixed points to be connected by a bridge and will try work out the most economical way of building the structure while keeping it safe and, ideally, visually pleasing.

In ancient times, Roman architect Vitruvian stated that a successful piece of architecture should be firm, functional and beautiful [20]. A few years ago, when discussing bridge design philosophy, [31] pragmatically modified the Vitruvian Triad and added a fourth principle: economy. With all these challenges and philosophy in mind, truss bridges have been designed and built by people for centuries [6].

Recently, computational optimization of truss structures has become a common research topic, with hundreds of publications every year [23] and impactful contributions from bio-inspired metaheuristic approaches, like evolutionary algorithms [29, 19] and particle intelligence [27, 26].

Weight minimization of trusses are usually main design goal of truss optimization studies, since lower masses make make structures safer under loading conditions [24]. These algorithms aim to find the best possible set of design variables regarding topology, shape and size of truss structures while fulfilling strength constraints [23]. Because bio-inspired algorithms rely on exploration and exploitation fundaments, it's possible to achieve global optimal solutions [17] and even push the barriers of knowledge by revealing previously unknown structure designs.

Truss structures have not not only been a recent common subject of scientific research papers, but, in popular culture, truss bridges have also been a recurrent subject of successful contemporary electronic games, like World of Goo [5], Poly Bridge [7] and Bridge Constructor [8]. These games, despite some artistic license, also apply the previously stated optimization and constraint satisfaction principles faced in real-world building of a truss bridge. Players are challenged, for example, to minimize material usage when creating a bridge that supports a given load without collapsing.

Having played many games of this genre during my childhood, the idea of writing a computer program that could generate optimized and beautiful truss bridges motivated me study this topic. Objectively, the creation of an evolutionary-based computer application for designing truss bridges can be of great help to architects and engineers, as they can provide insights, feasible structure designs and aid decision making [22].

1.2 OBJECTIVES

This project aims to create a computer application with an evolutionary algorithm that can generate and optimize planar truss bridges. The developed algorithm in this study focuses mainly on the minimization a truss structure mass, but also investigates the optimization of supported load and structural stress. The developed algorithm aims to simultaneously search a set of design variables concerning topology, shape and size of the bridge truss structure while optimizing a fitness function.

The output of the evolutionary algorithm is expected to converge and the generated trusses should also satisfy given functional requirements, like a stress limit for a given load when the structure is statically analyzed. Moreover, the developed application has the goal of being user-friendly, allowing its users to visually set the parameters of the structure to be generated and get visual outputs.

It is hypothesized that the generated output structures may showcase known architectural design traits, like symmetry and arches that result in a better structural stress distribution. At the same time, new organic shapes and other previously unknown patterns are expected be present as well, providing insights that can help users of the program in their design tasks.

1.3 STRUCTURE OF THE DOCUMENT

In Chapter 2, the theoretical basis of this report are discussed. Important concepts for this project, like trusses and evolutionary algorithms, are described. Furthermore, and a review of related literature is presented.

Chapter 3 focuses on the actual implemented program that generates and optimizes truss bridges. An overview of the program and data structures used to model the problem are provided, alongside deeper explanations of how the evolutionary algorithm and its operators were written. In the end of this chapter, the created graphical user interface is discussed and a website address for the open-source code base repository of the project is shared.

Chapter 4 shows a series of truss bridges generation experiments, with images of the resulting bridges and their structural analysis. Statistics and additional data about the evolutionary processes of the experiments are presented and discussed as well.

Chapter 5 presents the conclusions of this work and to which extent the initial objectives for the research were achieved. Limitations of the project and possible future improvements are also discussed in the last chapter.

CHAPTER 2

BACKGROUND

2.1 TRUSS STRUCTURES

Trusses are a type of structure that consist exclusively of straight members connected by joints located at their end points [3]. These structures are designed to support loads and are usually stationary. Trusses have been used since ancient times [6] and applications for planar trusses include not only bridges, but also roofs and cantilevers, for example.

Since early in the 19th century, many planar truss bridge designs have been developed for common use cases. Figure 2.1 shows some typical truss designs, like the Pratt, Howe and K-Truss patterns.

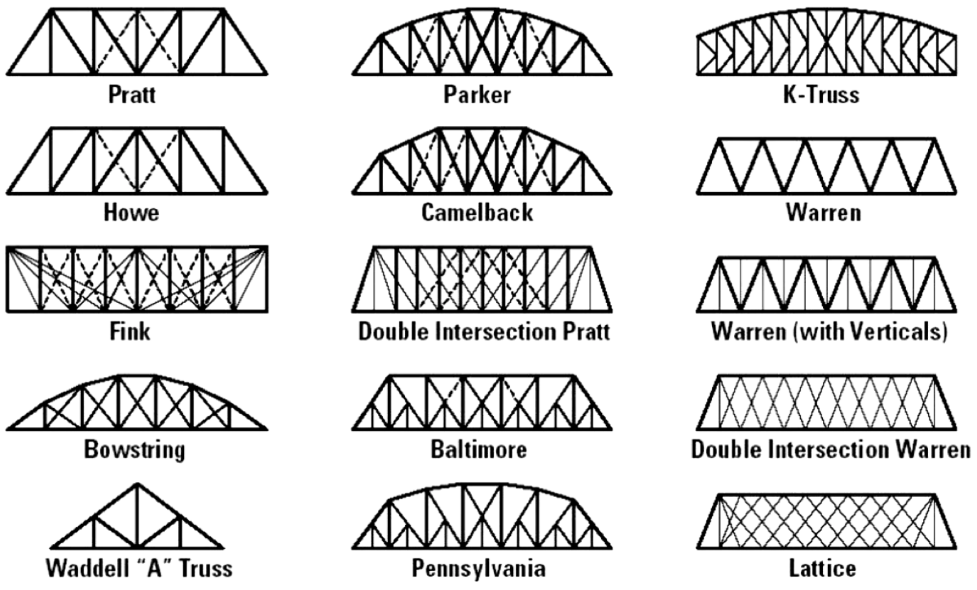


Figure 2.1 Common truss bridge designs [21].

Trusses can also be found on various bridges, including some that are not primarily labeled truss bridges. Figure 2.2 shows a view from underneath the the Golden Gate suspension bridge. Multiple truss structures can be seen, including its truss arch.

The analysis of trusses involve determining the forces in each truss member. After that, it is possible to discover nodal displacements and member stresses. This information is very important for evaluating truss designs and will be used to determine the fitness of a bridge, as discussed in Section 3.6.

Figure 2.2 View from underneath the Golden Gate Bridge showing its truss structures.

A few assumptions are made in order to allow further analysis and calculations of the forces inside the truss. Because members of a truss are slim and can support little lateral force, it is assumed that all loads are applied to the joints and not the members themselves. Moreover, it is reasonable to ignore friction on the pin joints, so the members are considered to be joined together by smooth pins. As a consequence of these assumptions, each truss member acts as a two-force member [13, 3].

There are many methods for performing a truss structural analysis and many parameters that need to be taken into account, like the location of the joints cross-sectional areas of the truss members. Other important parameters include material properties, such as its elasticity, density and strength. In this project, the Stiffness Method was chosen through an adaptation of the NuSA (Numerical Structural Analysis) framework [25]. According to [14], this method makes the formulation of matrices and operations easy and efficient for computer programs.

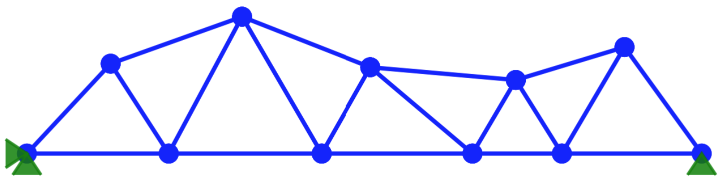


Figure 2.3 Sample truss bridge supported by two points.

Figure 2.3 shows an example of a truss bridge design. The green triangles in the image identify nodes with constrained movement, so they can be used as support points. The rightmost node is marked with one triangle pointing up, meaning that the node is constrained on the vertical axis, but is free to move horizontally. On the other hand, the leftmost node has two triangles, meaning that this node is fixed on both vertical and horizontal axes.

When designing trusses, the rigidity and stability of the structure must be guaranteed. For planar trusses, a minimum of three external support reactions (represented by the green triangles) are a partial requirement for truss stability [14]. With that in mind, it is common practice not to fully fix all support points from a bridge. Bridge bearings, represented by a lonely green triangle, give structural flexibility and allow controlled movement for the structure. This reduces structural stresses when dealing with thermal expansion, settlement of the ground below, seismic activity and other movement sources [18].

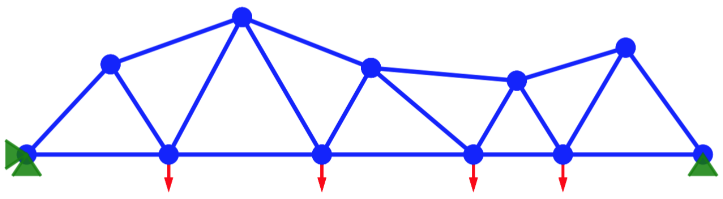


Figure 2.4 Sample truss bridge with load forces.

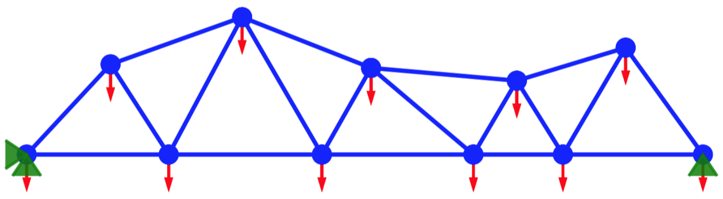


Figure 2.5 Sample truss bridge with weight forces.

Before submitting a truss bridge to analysis, a load force is distributed through the floor joints, as shown by the red arrows in Figure 2.4. Additionally, the weight of the truss members can be considered as well. To do that, a common practice explained by [13] is to divide the weight of each truss member equally into its extremities, as shown in Figure 2.5.

With input forces defined for the joints of a truss structure, it is possible to apply the Stiffness Method to get nodal displacements and, thus, member stresses. Figure 2.6 illustrates that by showing the original and deformed shapes of the bridge overlayed. In this image, stronger magenta colors mean greater tension forces on a truss member, while stronger cyan colors mean larger compression forces.

After the structural analysis, it can be determined if a truss structure can handle an applied load without any of its members reaching a stress limit or displacement limit, for example. The visual representation of the analysis also shows how the stresses are being distributed through the members so the bridge can be evaluated and, possibly, improved.

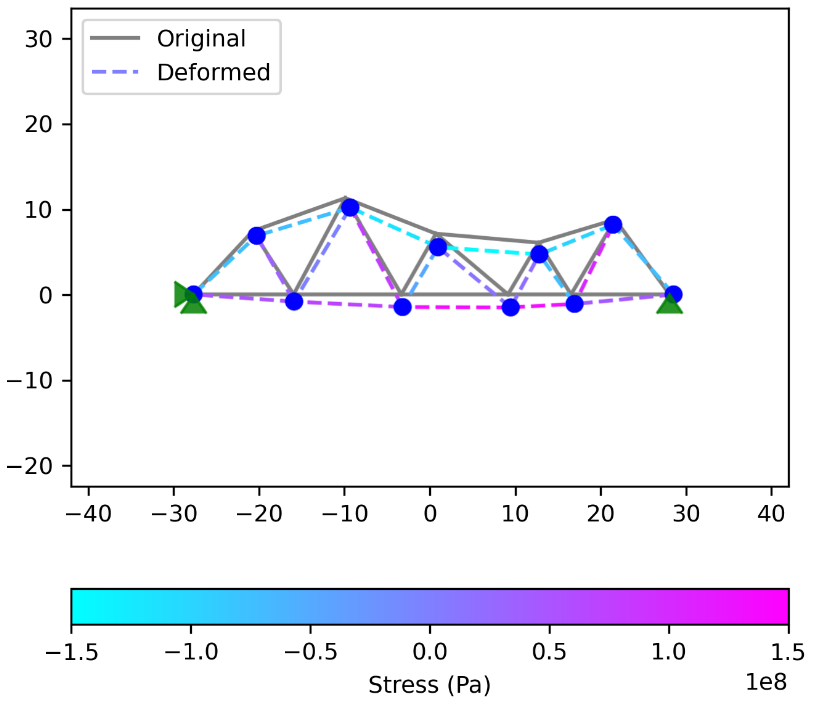


Figure 2.6 Analysis of sample truss bridge, with deformed shape and stress indicators after load and weight forces were applied.

2.2 EVOLUTIONARY ALGORITHMS

Evolutionary algorithms are a kind of metaheuristic algorithms inspired by natural biological evolution [17]. Although its origins can be traced as far back as the 1930s, it was during the 1960s that the popularization of computers allowed this field of study to flourish [15]. Currently, evolutionary computing is a major computer science field and is a very popular tool for optimizing complex problems solutions, specially where heuristics cannot be used [28].

The main ideia of this kind of algorithm is to solve a particular problem by having a set of candidate solutions, analog to a population of living individuals that want to survive and reproduce in an environment [10]. In the world, living creatures are constantly reproducing, mutating, and being naturally selected. When it comes to evolutionary algorithms, the set of candidate solutions will go through an analog evolution process [28]. By means of this trial-and-error approach, it is possible to achieve increasingly better solutions until a sufficiently good solution candidate is found [15].

There are many specific techniques for creating evolutionary algorithms. Broadly, this type of program will run a main loop, which represents the passing of generations of individuals. During each generation, the candidate solutions are evaluated through a fitness function. This function receives a candidate solution as input and outputs a number representing how fit is that solution. This allows individuals to be compared and selected. The best ones can be chosen for the next generation, for example, while the worst may be discarded [10].

During every generation, individuals stochastically suffer mutations and can reproduce to create new candidate solutions. This process is analog to Darwinian evolutionary concepts and the principle of survival of the fittest [15]. With evolutionary algorithms, complex solution spaces may be searched and solutions can be optimized, which can lead to solutions never though by humans before. Occasionally, the ideas behind these solutions are simple and understandable, as some examples found in Chapter 4. Sometimes, though, reasons behind solutions remain a mystery to be further investigated.

When commenting about about an astonishingly good structure design created by an evolutionary algorithm example, [17] stated: "For humans it looks very strange: it exhibits no symmetry, and there is no intuitive design logic visible. [...] It is a random drawing, drawn without intelligence, but evolving through a number of consecutive generations of improving solutions."

2.3 REVIEW OF RELATED LITERATURE

An overview and comparison of multiple research projects regarding the optimization of truss structures is provided by [23], which also presents an historic summary of the truss optimization studies and breakthroughs over the years. This article states that metaheuristic approaches, such as evolutionary algorithms, are a common choice for truss optimization because they are derivative-free, robust, simple to understand and its implementations are flexible.

Another comprehensive comparison of related projects is done by [24]. This article explains that traditional design objectives for truss optimization algorithms that use static analysis are mass, displacement and stresses. With one or more of these design goals, truss optimization algorithms aim to find the best possible set of design variables such as topology, shape, size or a combination of them.

One example of evolutionary approach is given by [19], which proposes a hybrid evolutionary firefly algorithm for shape and size of truss structures. This article focuses on changing the positions of nodes and the cross-sectional areas of members from a given truss structure. Interestingly, when submitting a 37-bar planar truss that follows a Pratt pattern to shape evolution, the optimal outcome was an aesthetically pleasing arch, similar to the ones found by previously published works.

While [19] and similar works bring advancements to the truss optimization with evolutionary algorithms field of study, these articles do not change the topology of the structure during the evolution, like it is proposed on this project. Therefore, the initial population is not procedurally generated and the end results always remain with the same members and nodes from the first individuals.

On the other hand, [22] uses multiobjective evolutionary algorithms to simultaneously optimize topology, shape and sizing of plane trusses. This approach allows creation of new truss members and nodes, while also varying cross-sectional areas and nodal positions. Results from this study show that these design strategies are efficient and effective. The authors of the article write that the proposed design approach can be a powerful engineering design tool, since it provides feasible truss structures that can help decision making. Yet, the project does not focus on bridges examples nor mentions the creation of a user interface for an engineering tool.

When it comes to the history of algorithms for optimizing truss structures, genetic algorithms were very common since the beginning of these studies and [16] is an example of that. It describes a truss topology optimization algorithm and makes fundamental advancements regarding minimization of invalid truss structures during the evolutionary process, such as avoidance of needless members or those which overlap other members. The proposed approach also guarantees that the individuals of the population are always stable, a feature that drastically improved the performance of their algorithm.

As stated by [10], evolutionary algorithms are analogous to a trial-and-error approach. Given that the search space for a truss structure design is virtually infinite, evaluation of known bad designs can be a huge problem for evolutionary algorithms and should be avoided whenever possible.

Other articles, like [30], provide basis for the creation of differential evolution approaches to truss optimization algorithms. The evolutionary loop and all of its operators are described in the context of trusses and [9] also applies similar evolutionary strategies in its development. These articles both use genetic algorithms and focus on the minimization of weight given design constraints, providing insights to the developments of the project described in this thesis.

CHAPTER 3

PROJECT OVERVIEW

3.1 TRUSS MODELING

In order to apply evolutionary operators to trusses, this type of structure was modeled as an undirected graph. In this context, truss joints and members were represented by graph vertices and edges, respectively.

This modeling was done following an object oriented pattern. A class for vertices was implemented with properties such as position, movement constraints and applied force during analysis. Another class for edges was also created containing properties like as its vertices, material elasticity and cross-sectional area. Computed properties, such as edge length and volume were also added. These computed properties, alongside material density information, allow the calculation of the mass from a given truss member.

With vertices and edges building blocks, it was possible to create a graph type that represents a truss using the adjacency list approach [4]. The graph objects gained the ability to add, remove and modify its vertices and edges. These methods will be the basis for the evolutionary operators that modify the truss bridges structures, such as mutations and crossover.

3.2 TRUSS RIGIDNESS

Functions to check rigidness and stability were added to the truss graph class, so that it was possible to know if a truss structure was rigid or if it would collapse, for example. One quick check implemented to verify instability is $b + r < 2j$, for a truss with b bars, r reactions and j joints [14]. If the expression is true, the planar truss will surely be unusable. On the other hand, if this check passes, then other tests need to be performed before guaranteeing stability.

Another more comprehensive method for checking truss rigidness was implemented using the stiffness matrix of the structure [14]. This function is able label a structure as stable or not for sure, but with a higher computational cost than the previously shown test. Because we assume the truss members to be linearly elastic, the Maxwell's reciprocal theorem applies and tells us that a displacement produced at any point A due to certain load applied at point B should be equal to the displacement produced at point B when same load is applied at Point A [12].

If a stiffness matrix has a non-zero determinant, it can have its inverse computed, and we can be sure the truss associated with the matrix is well behaved. Otherwise, if the matrix is non-revertible, we can say it is ill-conditioned. This means that the structure is not stable because it has an exploding condition number.

3.3 BRIDGE GENERATION

Unlike the majority of related studies, the individuals of the initial population of truss bridges feature randomness and are not all equal. To do that, a procedure was developed for generating valid truss bridges. The inputs of this bridge generation function are at least two floor support positions and, optionally, extra anchor points arbitrarily positioned. Other global parameters are also used throughout the program, like the minimum distance between any two vertices and the maximum length for an edge. The interval between these two constants is referred in this work as a vertex connection range to another vertex.

The function for creating a bridge follows 3 main steps. The first one is about connecting all the floor anchor points with a line of edges of random lengths, following a normal distribution, as shown by an example in Figure 3.1. Then, the function proceeds to the second step and connects each of the extra support points to the closest existing floor vertex with a line of edges, as seen on Figure 3.2. This gives us a structure of lines connecting all given support vertices which is not rigid yet.

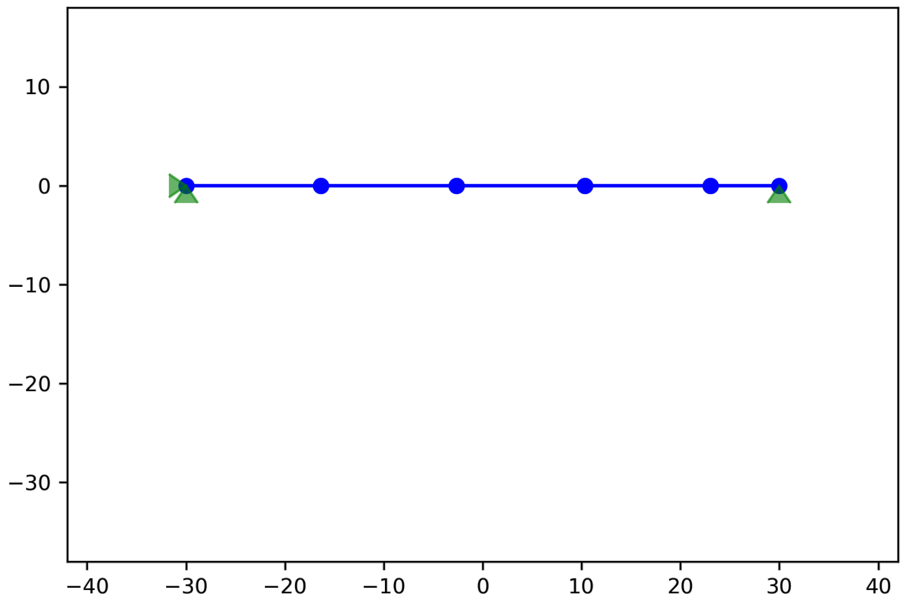


Figure 3.1 Example of the first step of bridge generation.

When this inicial structural skeleton is complete, the third step starts. The goal of this step is to create triangles and it is inspired by the Warren truss design (Figure 2.1). For every pair of neighboring vertices in the original skeleton structure, an equidistant new point is randomly is added, forming a triangle. During this process, if a new vertex is within the established connection range, they are also connected. This step is shown in Figure 3.3. Note that, for each line from the original skeleton, new vertices are added to only one side of the line. This side is randomly chosen for every line and another example

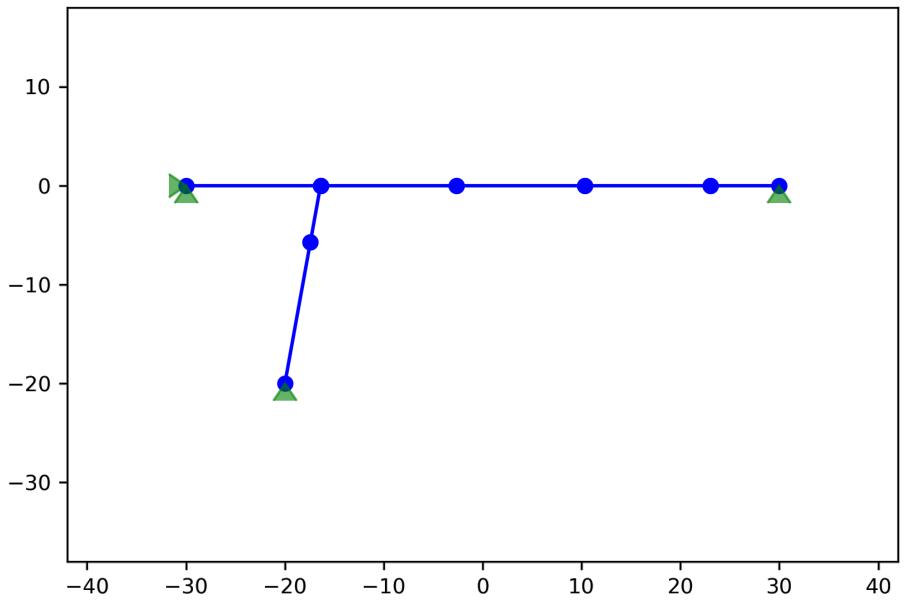


Figure 3.2 Example of the second step of bridge generation.

of a bridge generated for an initial populations is shown by Figure 3.4.

Usually, after the third step, the truss structure is already rigid and complete, like the examples referenced in this section. But, if the bridge is not rigid after the third step, a fourth step takes place. It adds new randomly positioned vertices and connects them to existing vertices within the connection range until the bridge is stable.

This method guarantees the generation of rigid truss bridges, which prevents the overhead of dealing with invalid individuals later in the program execution. Moreover, the generated bridges show great variability, which helps the exploration of solutions during the evolutionary process. Because of the Warren design inspiration, these generated bridges already make some structural sense, which increases the fitnesses of the initial bridge population.

3.4 MUTATION

The developed evolutionary algorithm has multiple mutation functions that can be applied to a truss bridge. The main goal of these functions is to allow changes to the structure but still keep it rigid and valid. For example, when removing an edge from a truss, we must guarantee that the bridge still has a continuous floor path and that it will not collapse. These mutation methods cover topology, shape and sizing of the structure by taking advantage of the graph model for a truss.

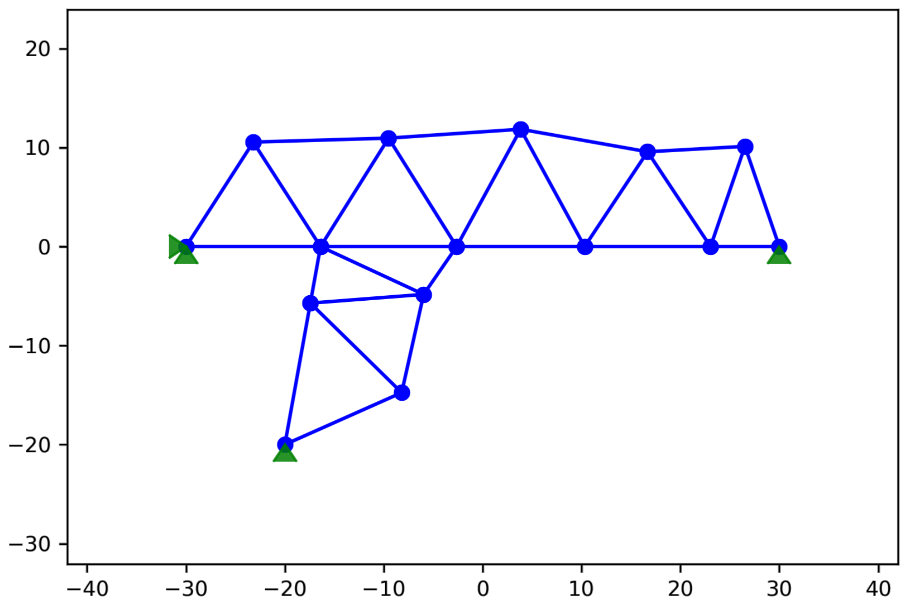


Figure 3.3 Example of the third step of bridge generation.

3.4.1 Topology

The implemented topology mutation operations regard removal and addition of random edges and vertices. Edge removal is done by randomly choosing an edge and checking if the edge can be removed without compromising the rigidness nor the floor of the bridge. If the check passes, the edge is then removed. Otherwise, other random edges are checked until the operation succeeds or a tryout limit is reached. This approach of limited trials is repeated on all of the evolutionary operations that have a chance of failure in this project.

To add an edge, a random vertex from the graph is chosen. Then, all vertices within connection range to the first vertex that are not already connected to it are gathered into a set. Finally, a random vertex from this set of possible vertices to be connected is selected and the new edge is established.

The process of vertex removal starts by randomly selecting a candidate vertex to be removed that is not an anchor point. If the selected vertex is not part of the bridge floor, we check if the truss would still be rigid without the vertex. If the vertex can be safely removed, then it is removed and the operation is completed. On the other hand, if the vertex is part of the bridge floor, it is attempted to remove that vertex and connect the neighboring floor vertices in order to maintain a continuous floor line.

Adding a vertex to the truss bridge begins by randomly selecting a vertex from the existing structure and finding a random position within connection range to it. Then it is attempted to add a vertex at that location, making sure that the new vertex can be connected to at least two existing vertices. Also, it is assured that the new vertex is not closer to another vertex than the established minimum vertex distance. As an

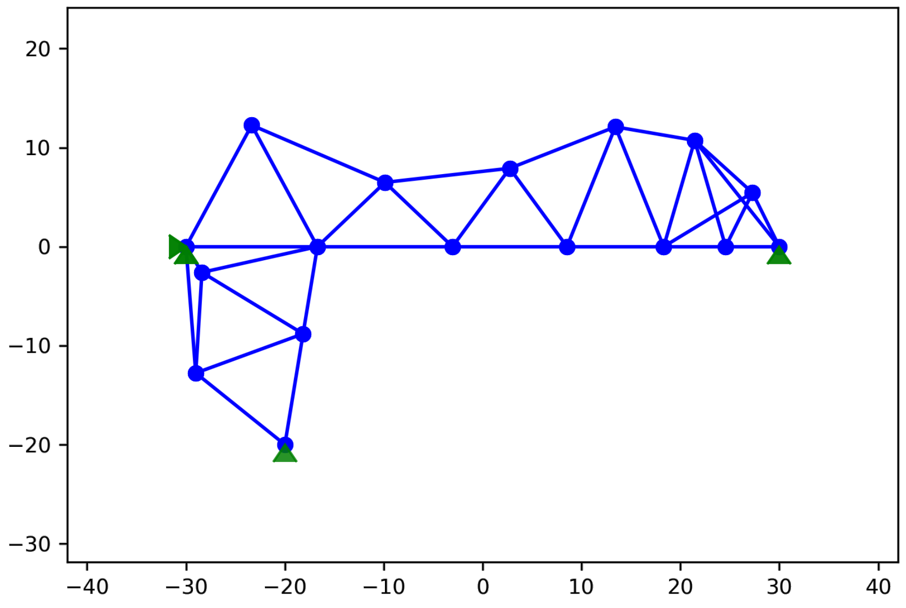


Figure 3.4 Another example of a bridge generated for an initial population.

optimization, if the new vertex would be arbitrarily close to another edge, the program was designed to split the existing edge into two to prevent overlaid edges.

3.4.2 Shape

In order to change the shape of the structure, a function to move vertices was created. To do that, a random vertex is chosen and its location shifted by a random amount following a gaussian distribution. If all of the existing edges of the vertices still have valid lengths and the new vertex position is not closer to other existing vertices than allowed, the operation is condered successful.

3.4.3 Sizing

A function to change de cross-sectional area of the truss members was implemented respecting constants for maximum and minimum areas. The method chooses a random edge from the truss graph and changes its existing cross sectional area by a random amount following a gaussian distribution.

3.5 CROSSOVER

The crossover procedure combines two trusses by copying a piece from one truss graph and pasting it into another structure. It was inspired by [11] and the process starts by randomly selecting a vertex from one of the graphs. Then, a random radius is generated in order to determine a circular area around that random vertex. All the vertices and edges encompassed by the circle on the first graph are copied into a temporary variable.

Then, on the second graph, all the vertices inside the same circular area are removed, including their edges. The temporary variable subsequently pastes the vertices and edges copied from the first graph into the second one, making new edges as needed. If the resulting structure is stable, the operation is finished.

3.6 FITNESS FUNCTIONS

Evaluating how fit is an individual is a core part of any evolutionary algorithm. For this project, it was decided to focus on three properties of a truss bridge: structural mass, structural stress and supported load. By fixing two of the three parameters, it is possible to optimize the third, as shown in the following subsections.

In order to measure structural stress and, consequently, supported load for a truss bridge, the Stiffness Method was chosen through an adaptation of the NuSA (Numerical Structural Analysis) framework [25].

3.6.1 Minimizing Material

A fitness function that measures and outputs structural mass was created so the minimization of material could be achieved with the evolutionary algorithm. As long as a candidate bridge supports a given load without any of its members reaching a given stress limit, it is reasonably strait forward to measure the mass of the structure. To do that, all members volumes are calculated by multiplying their length by their cross-sectional areas. Then, the total volume is obtained and multiplied by the truss material density, which results in the total bridge mass. If the candidate bridge does not meet the load or stress requirements, it suffers a fitness penalty and it is flagged to indicate that it is not viable.

When dealing with mass, or material usage, a smaller number means better fitness. But, for implementation consistency, it was chosen to use the inverse of that number, so a greater fitness always means a better individual on all fitness functions implemented in this project.

3.6.2 Minimizing Structural Stress

Given a fixed load and a structural mass limit, it is possible to optimize structural stress. First, it is assured that the candidate bridge supports the applied load without exceeding the stress limit for any of the truss members and without exceeding the structural mass limit. Then, the fitness function can output a value for the structural stress of a bridge. There are a few approaches to do that, and in this project it was chosen to take the mean of a few of the most stressed members from the truss. Again, a smaller stress number means better fitness. But, for implementation consistency, it was chosen to use the inverse of that number in this project.

It is interesting to note that, for this fitness function, that the mean of all truss members from a bridge will not achieve the desired result. When using the mean stress, an evolutionary algorithm may start to create a huge number of nonsensical members attached to an anchor point. These members will tend to have a low stress because they are attached to a fixed point. This will create a structure with a small stress mean, but the bridge will not make much sense nor achieve the desired outcome.

3.6.3 Maximizing Supported Load

To measure the maximum supported load, we first set maximum limits for structural stress and structural mass. Because the Stiffness Matrix Method of analysis, used in this project, only gives us the structural stress for a fixed load, a method similar to a binary search was used to find the maximum supported load before the bridge collapses. To do that, an arbitrary precision is set and a series of tests are carried out until it is possible to assure that the maximum supported load of a bridge is within the given precision. Finally, this value for maximum supported load is outputted.

3.7 EVOLUTION LOOP

For the purpose of performing the evolution process, an evolution loop was created which contains the core of the developed program. An adaptation of the Evolution Strategies [10] algorithm was chosen and this loop applies the bridge generation, mutation, crossover and fitness operations, allowing the population to evolve through multiple iterations.

First, parameters like the number of individuals, evolution generations limit and replacement percentage – a form of selection pressure – are given. Then, the initial population of bridges is created according to the specifications for anchor points and extra supports. To finish the setup phase, the population is sorted by their fitnesses.

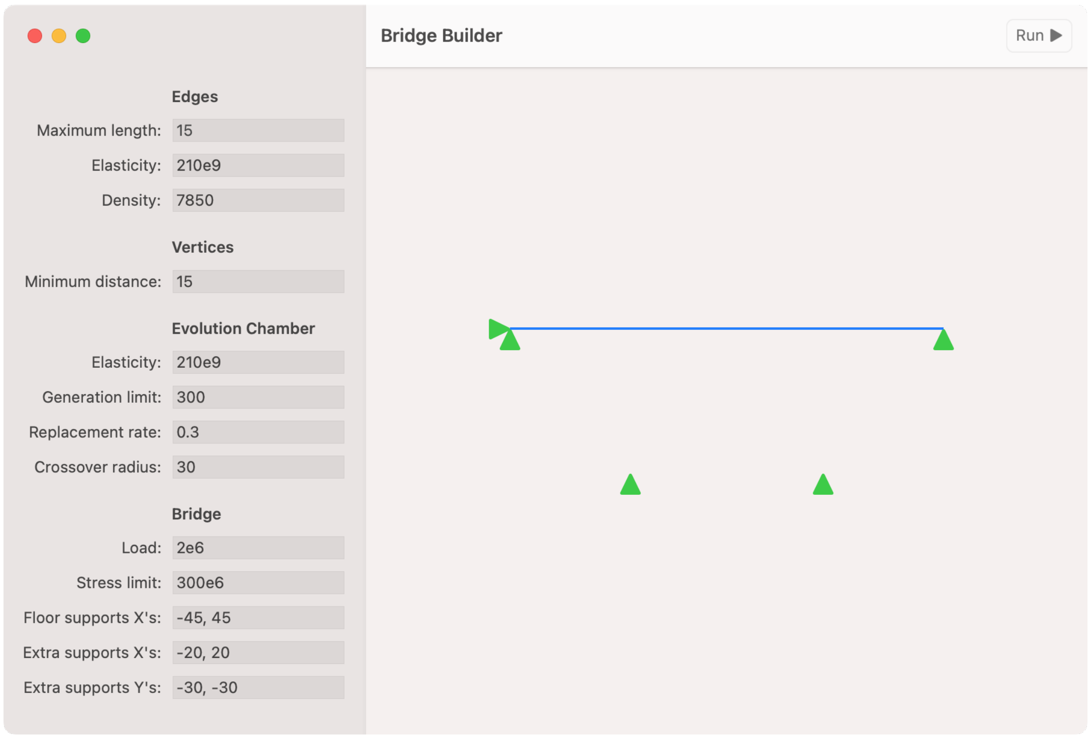
After that, the actual loop starts. For every generation, the population suffers a random amount of mutation operations, following a gaussian perturbation when possible. Next, a percentage of the population, is replaced using an elitist (μ+λ) approach. Parents selection follows a uniform random method within a group of best fit individuals.

Throughout the whole evolution loop, the best individuals and various details about the population are stored. This allows the generation of statics, plots and other insights, as shown in Chapter 4.

3.8 USER INTERFACE

One of the main goals of this project is to help engineers and architects with decision making in their tasks to design truss bridges. To accomplish that goal, it is very important for the developed system to be user-friendly and easy to use. Because of that, an intuitive, responsive and user-centered [2] graphical user interface was developed, as shown in Figure 3.5.

On the left, the user interface window features a column of text fields that allow easy editing of the parameters of the bridge. On the right, there is a real-time drawing of the floor of the bridge and its support points. This allows a clear visualization of the bridge support points and gives an idea of the shape of the bridge that will be generated. Finally, on the top right there is a "Run" button. When pressed, the program starts its evolution loop in order to minimize structural mass. The program also saves plots and other data about the best fit bridges into a predetermined file system folder.



Edges

Maximum length:	15
Elasticity:	210e9
Density:	7850

Vertices

Minimum distance:	15

Evolution Chamber

Elasticity:	210e9
Generation limit:	300
Replacement rate:	0.3
Crossover radius:	30

Bridge

Load:	2e6
Stress limit:	300e6
Floor supports X's:	-45, 45
Extra supports X's:	-20, 20
Extra supports Y's:	-30, -30

Figure 3.5 Developed graphical user interface for the developed system.

3.9 SOURCE CODE

All the source code for this project, including documentation and code comments, are available on the ”BridgeBuilder” public repository at `https://github.com/HugoLis/BridgeBuilder`.

CHAPTER 4

EXPERIMENTS

Some experiments were carried out in order to showcase the capabilities of the developed system in different situations. To run the experiments, a base model 16-inches MacBook Pro computer from 2019 [1] was used. All the experiments focus on the minimization structural mass, since this is one of the main real-world goals of truss optimizations. This means that the program aims to find the truss bridge with the least mass that can support an applied load without collapsing.

Some edge parameters remained constant throughout the experiments, as shown in Table 4.1. These parameters include material properties and the cross-sectional area of the truss members. It was decided not to apply sizing mutation operations, so that the number of variables could be reduced in these initial experiments with the developed application.

The chosen material for the truss members was a common kind of structural steel, used in many types of buildings. The exact values of the material properties are not extremely important to the experiment, as long as their orders of magnitude are somewhat realistic. Other parameters were changed for each experiment, like applied load, population size and the support points for the bridge to be generated.

Material Elasticity	210GPa
Material Density	7850 Kg/m^3
Material Yield Strength	300MPa
Maximum Crossover Radius	30m
Member Maximum Length	15m
Member Cross-Sectional Area	1cm
Minumum Vertices Distance	3m

Table 4.1 Constant parameters during the experiments.

4.1 SINGLE-SPAN BRIDGE

Single-span bridges are probably the simplest kind of truss bridge and there are many common designs for them, as shown in Figure 2.1. In this experiment, only a couple of floor support points were given with no extra anchor points. The parameters of this experiment are shown in Table 4.2.

In a total of 300 generations, the individual with least material that could support the applied load without collapsing was achieved on generation number 284. The resulting bridge is shown on Figure 4.1 and features an arch along the top of the floor, similar

Population Size	80
Generation Limit	300
Applied Load	2.5MN
Floor Support Points	(-30, 0); (30, 0)

Table 4.2 Parameters for the single-span bridge experiment.

to a Bowstring truss design (Figure 2.1). The member of the structure with maximum stress had a stress value of 292MPa, which is over 97% the given material strength value. Figure 4.3 shows the analysis for this bridge and it is noted that the distribution of stress appear to be very uniform along the floor members (under tension) and members of the arch (under compression).

Table 4.3 shows some statistics about the evolution process over the generations and indicates that the program converged, at least to a local maximum. In the fitness mean graph, it is observed a gradual growth with a couple of moments around generations 100, 200 and 300 in which the growth appear to stagnate. This probably happened because the population was stuck in local maxima. The best fitness plot shows us a mostly linear growth over time and the standard deviation graph indicates that the best individuals fitnesses became very uniform towards last 100 generations.

It is important to note that every program execution may lead to different results. On another run of the program, with the exact same parameters, an alternative solution was found with a very similar fitness. This alternative solution is shown on Figure 4.2 and it features the same arch structure found on Figure 4.1, but below the floor level. In this alternative example, unlike the original solution, the floor members are under compression while the arch members are under tension.

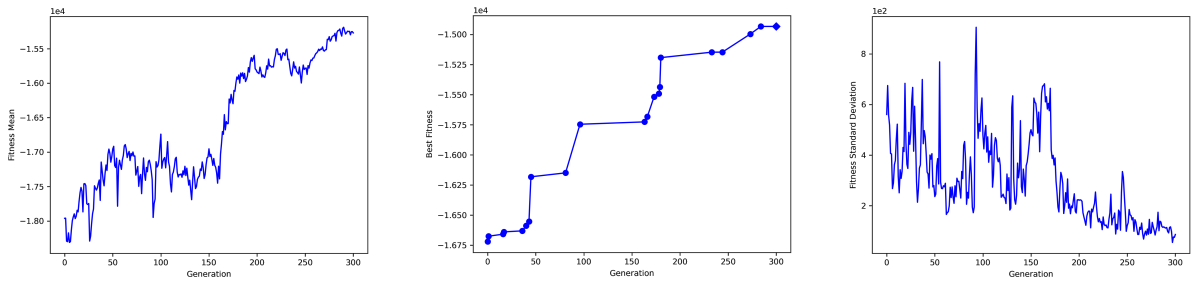


Table 4.3 From left to right, statistics throughout generations of the single-span bridge experiment for best fitness, fitness mean of the top 20% of the population, and fitness standard deviation of the top 20% of the population.

4.2 DOUBLE-SPAN BRIDGE

After experimenting with a simple bridge with two supports on its extremities, the natural next step was to run the program in a double-span bridge context. In practical terms,

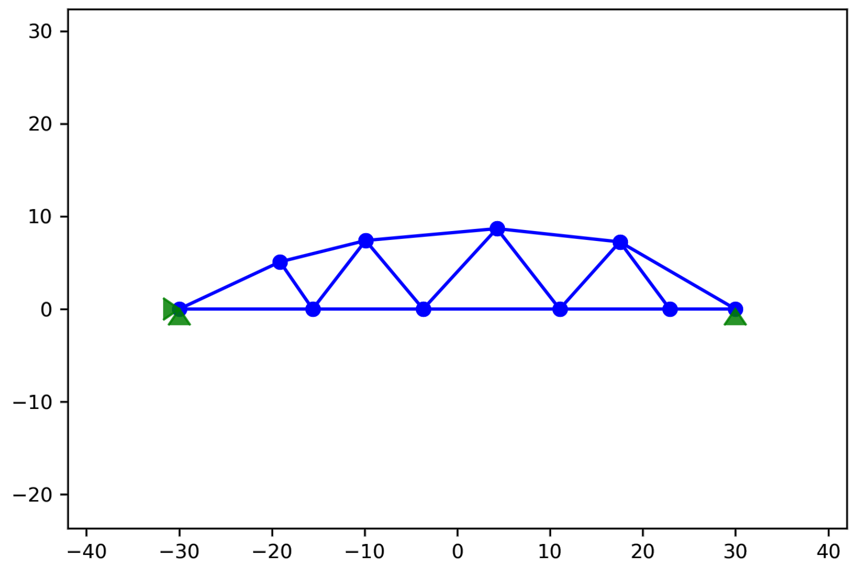


Figure 4.1 Best single-span bridge, achieved on 284th generation.

this means adding one extra floor support point to the structure. The exact parameters for this experiments can be seen in Table 4.4.

Population Size	120
Generation Limit	150
Applied Load	6MN
Floor Support Points	(-30, 0); (0, 0); (30, 0)

Table 4.4 Parameters for the double-span bridge experiment.

Table 4.5, in the upper-left, shows an example of a random bridge from the initial population of this experiment. The image shows some overlapping edges and vertices that appear too close, which are not good indicators for a well-performing truss bridge. Of course, this is expected, since this individual did not go through the evolutionary process. The subsequent images on the table exhibit, from top to bottom and left to right, some of the best fit individuals during the evolutionary process. They are, respectively, the best fit bridges from generations 17th, 20th and 60th.

It is interesting to note that the program considered an arch, like the one from the single-span bridge experiment, during its 20th generation. But, after the generation number 60, the final topology for the best individual of the experiment was already established. After some minor shape mutations, the individual with least mass that could support the applied load was found on generation 106. This best fit bridge is illustrated in Figure 4.4 and it features an unusual, but, to some degree, symmetric shape. While its left portion features a structure above the floor level, the right portion features a similar

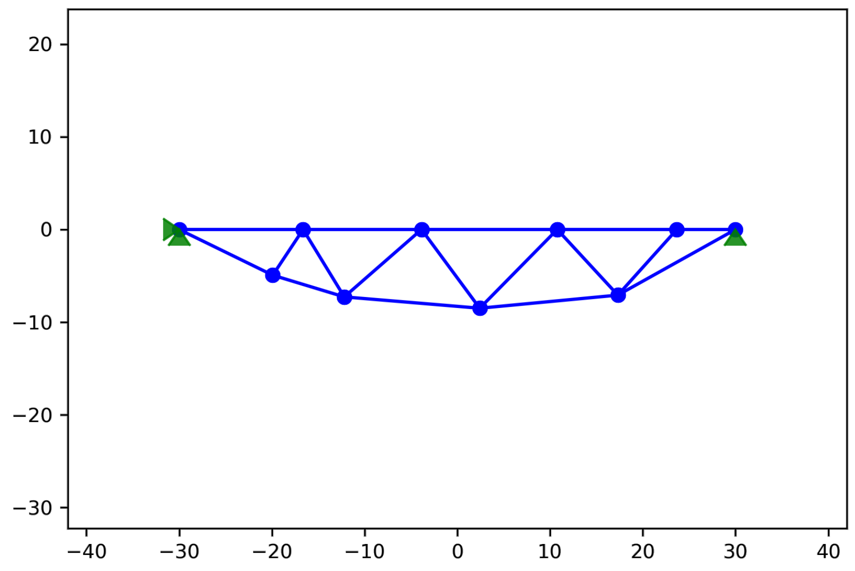


Figure 4.2 Best single-span bridge alternative, achieved on the 282nd generation.

structure below the floor level.

The stress value of the most stressed truss members from the best bridge was 297MPa, which is 99% of the maximum allowed stress before material breakage. This an indication that the structure is indeed optimized. This assumption is also supported by the analysis of the bridge, on Figure 4.5. The structure exhibits most of its members very close to the stress limit, which are strongly colored in cyan or magenta.

Additional data to the double-span bridge experiment is provided by Table 4.6. In the plots, it is possible to see that the best fitness and the fitness mean grew very fast in the beginning, but, became slower near the end. The standard deviation of the best individuals also featured a compatible shape, starting high and getting lower near the final generations. This behavior suggests that the experiment converged, which is the desired outcome for evolutionary algorithms.

4.3 BRIDGE WITH ASYMMETRICAL PIER

The first two experiments were set in a symmetrical environment with regards to the support points of the bridge. With that in mind, the third experiment adds an asymmetrical pier in the lower left quadrant of the structure frame. The parameters used in this experiment are found in Table 4.7.

In the top-left of Table 4.8 it is possible to examine a random sample from the initial population of this experiment. The bridge appears to have a lot of unnecessary edges and vertices and, for sure, has a horrible structural mass fitness. The bridge in the top-right of the table, on the other hand, is much cleaner. It does not feature overlapping edges nor vertices very close together. This bridge is the best fit of the the first generation.

The design continues to be improved over the generations. In the bottom-left of

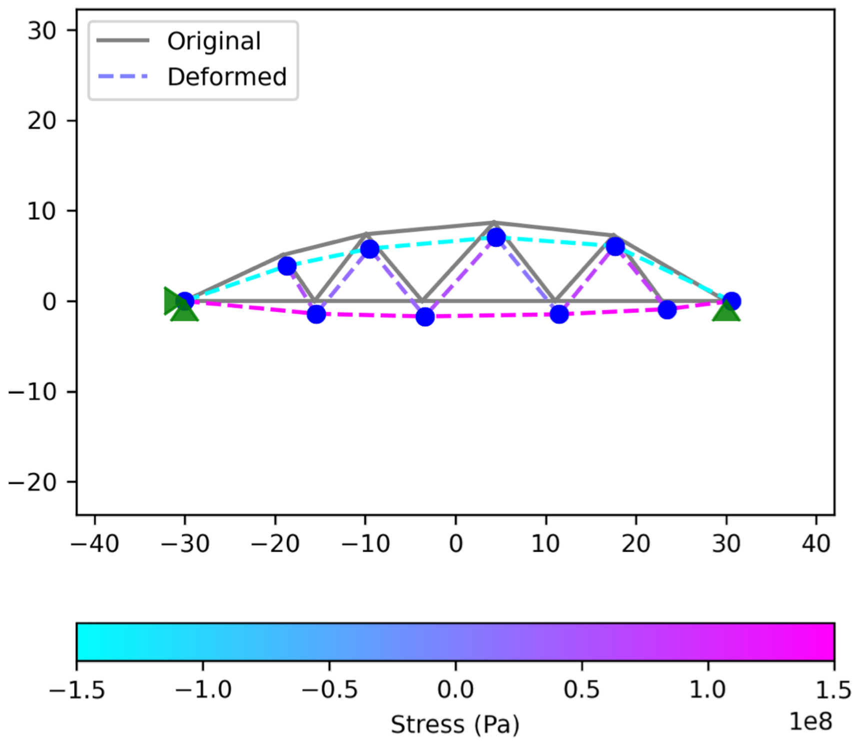


Figure 4.3 Analysis of best achieved single-span bridge.

Table 4.8, we see the best bridge from the 19th generation with an interesting design. It features an arch on the right and a wedge-shaped column on the left. The bridge is further simplified and the best bridge from the 200th generation is shown on the bottom-right of the table.

The best fit bridge for this asymmetrical pier experiment was found during generation 316th and is shown on Figure 4.6. It features a simple and curious design, with a triangular column on the left and an arch that resembles a Parker truss (Figure 2.1) below the floor level on the right. The most stressed member of this bridge reached a stress of over 299MPa, at almost 100% of the maximum 300MPa allowed before we consider that the bridge collapsed. This suggests that the bridge was highly optimized.

The level of optimization is also confirmed by the analysis of the bridge, illustrated by Figure 4.7. There, we see stress levels very near the limit and also very well distributed across the members of the truss.

The plots provided by Table 4.9 also give us important insights and indicators of convergence on all curves. The best fitness and best fitness mean graphs feature rapid grow during the initial generations and tend to a constant limit near the end. Additionally, the standard deviation plot starts very high and becomes lower as generations pass.

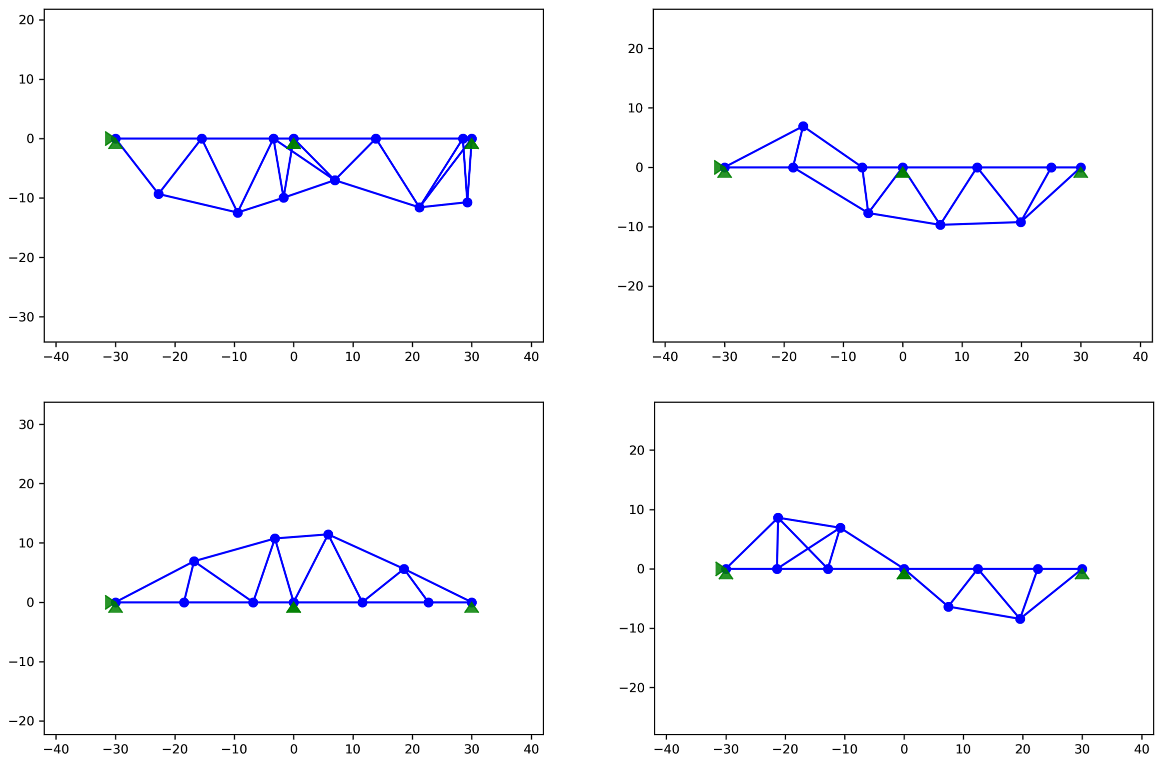


Table 4.5 From top to bottom and left to right, a random generated bridge during the double-span bridge experiment and the best bridges during the 17th, 20th and 60th generations, respectively.

4.4 BRIDGE WITH TWO PIERS

The fourth and final experiment symmetrically adds two extra supports in addition to the two floor supports. With a total of four supports, this experiment is the most complex and expensive regarding computer resources out of the ones written about in this report. The parameters used for the bridge with two piers experiment can be found in Table 4.10.

After 200 generations, the best fit bridge was created during the 190th generation. The resulting bridge is shown in Figure 4.8 and it features two mostly symmetrical columns that attached to the bottom supports. These columns are very thin to reduce mass and they connect to all floor vertices. The most stressed truss members had a stress value of 233MPa. This represents a little bit less than 78% of the maximum allowed stress, which may indicate that further improvements could still be done to the structure.

The analysis of the bridge, illustrated in Figure 4.9, shows multiple members with strong colors and others, like the central floor member, with almost zero stress. Table 4.11 shows us insightful plots, with a very rapid fitness increase up to the 25th generation and then a very slow growth afterwards. The fitness standard deviation plot shows decrease. These statistics indicate that the program converged, at least, to a local maximum during this experiment.

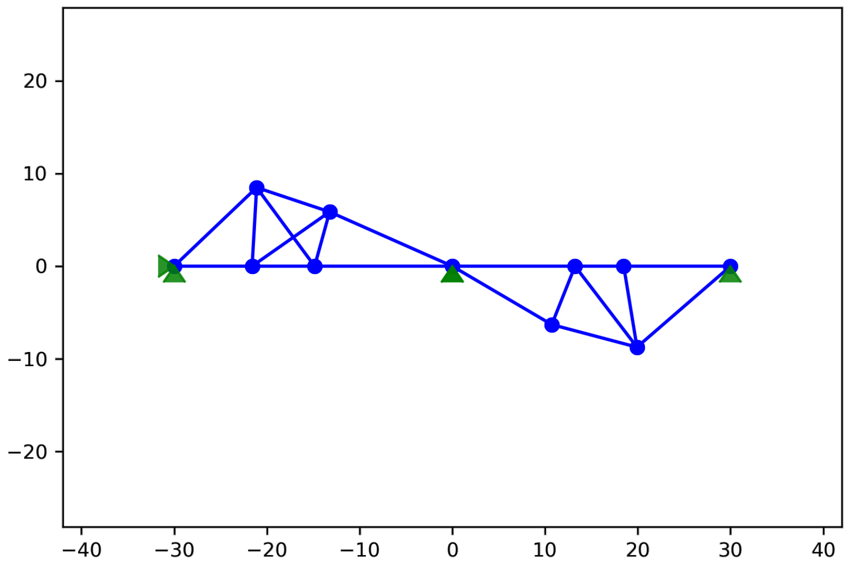


Figure 4.4 Best double-span bridge, achieved on the 106th generation.

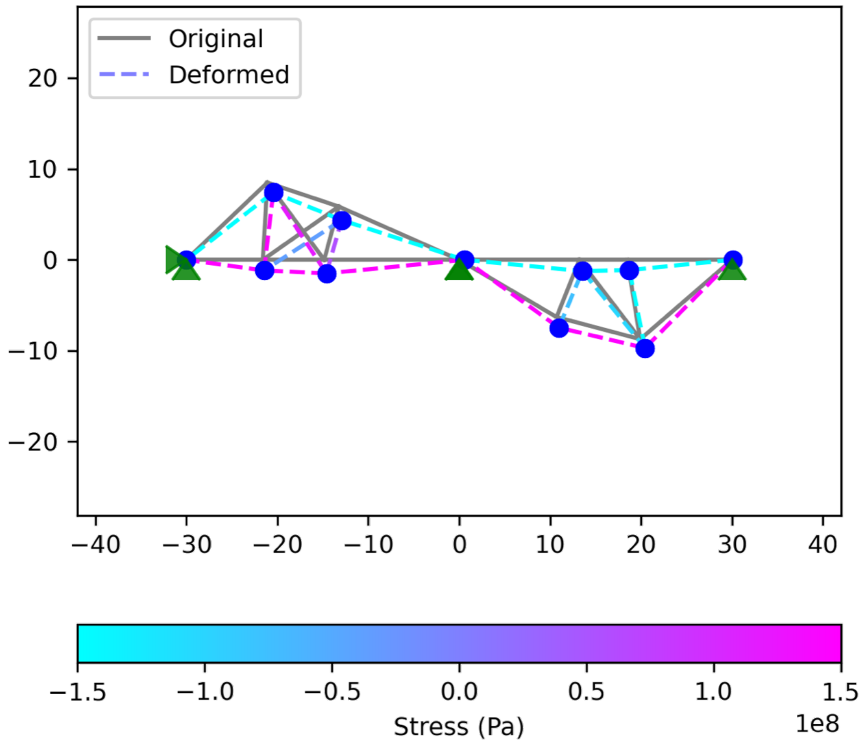


Figure 4.5 Analysis of best achieved double-span bridge.

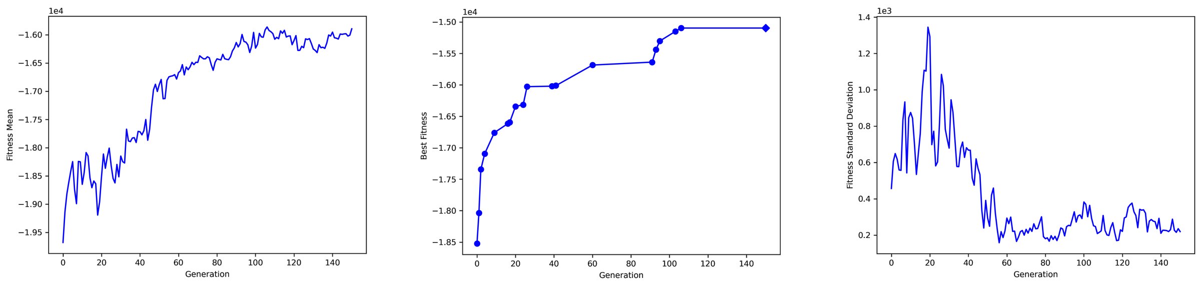


Table 4.6 From left to right, statistics throughout generations of the double-span bridge experiment for best fitness, fitness mean of the top 20% of the population, and fitness standard deviation of the top 20% of the population.

Population Size	120
Generation Limit	350
Applied Load	4MN
Floor Support Points	(-30, 0); (30, 0)
Extra Support Point	(-15, -25)

Table 4.7 Parameters for the bridge with asymmetrical pier experiment.

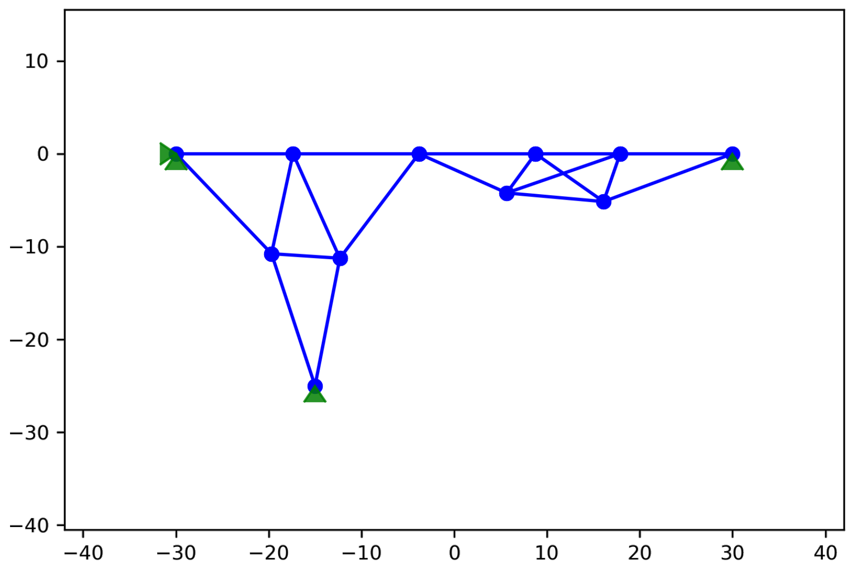


Figure 4.6 Best bridge with asymmetrical pier, achieved on the 316th generation.

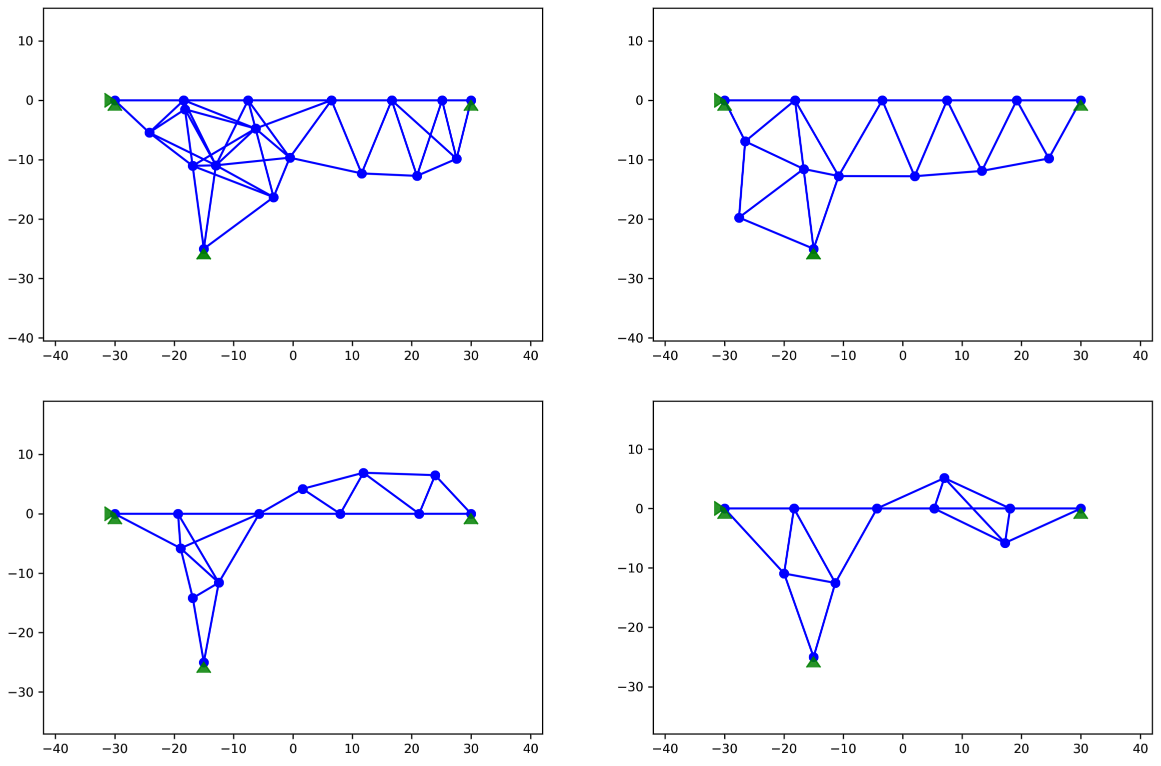


Table 4.8 From top to bottom and left to right, a random generated bridge during the bridge with asymmetrical pier experiment and the best bridges during the 1st, 19th and 200th generations, respectively.

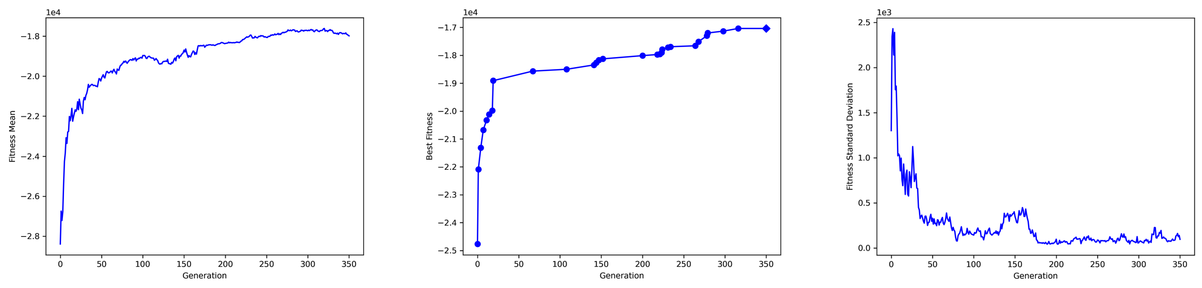


Table 4.9 From left to right, statistics throughout generations of the bridge with asymmetrical pier experiment for best fitness, fitness mean of the top 20% of the population, and fitness standard deviation of the top 20% of the population.

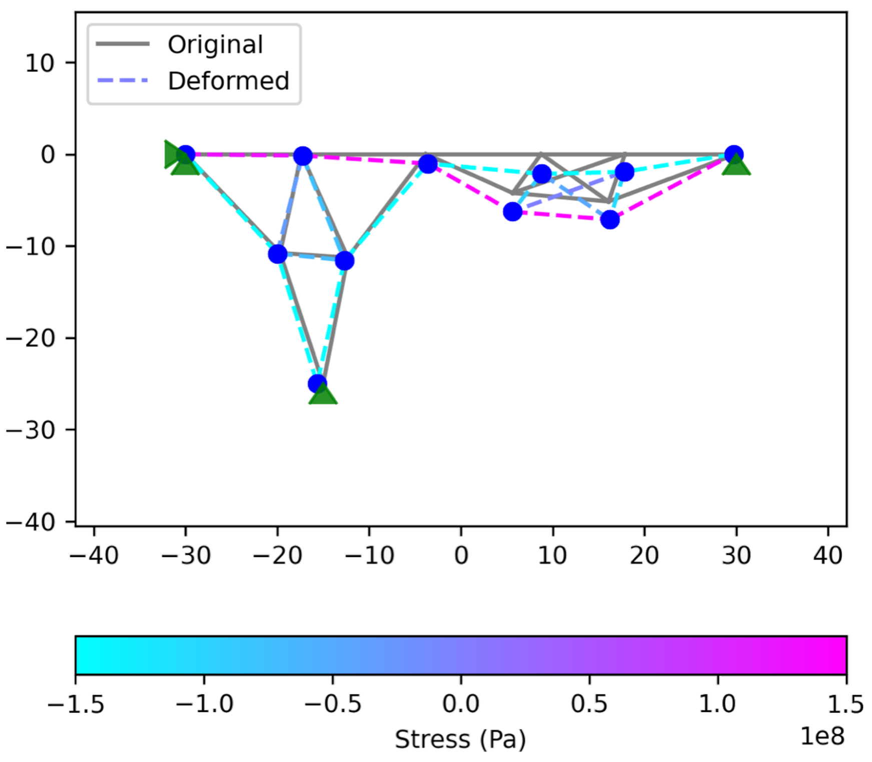


Figure 4.7 Analysis of best achieved bridge with asymmetrical pier.

Population Size	120
Generation Limit	200
Applied Load	6MN
Floor Support Points	(-30, 0); (30, 0)
Extra Support Points	(-15, -20); (15, -20)

Table 4.10 Parameters for the bridge with two piers experiment.

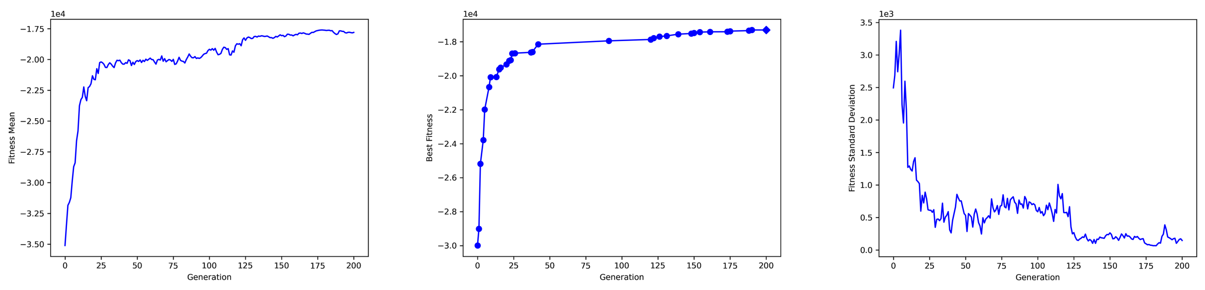


Table 4.11 From left to right, statistics throughout generations of the bridge with two piers experiment for best fitness, fitness mean of the top 20% of the population, and fitness standard deviation of the top 20% of the population.

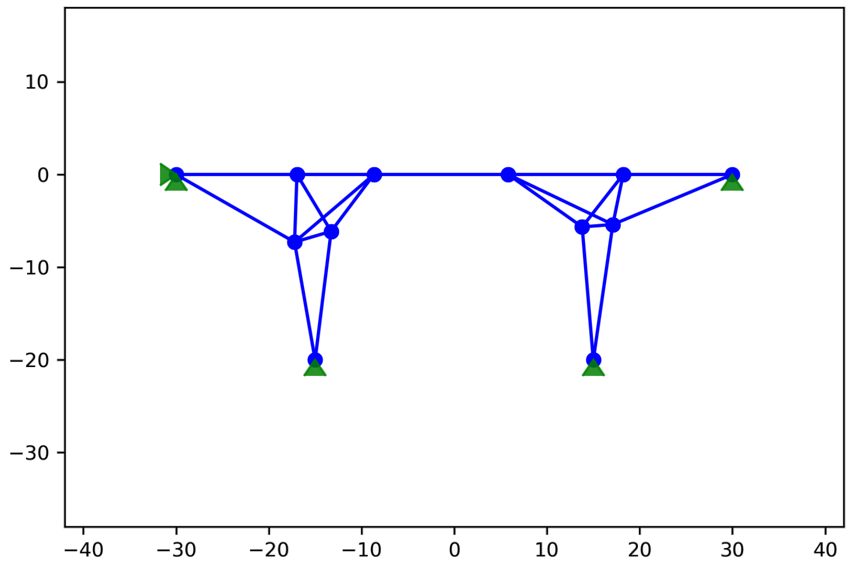


Figure 4.8 Best bridge with two piers, achieved on the 190th generation.

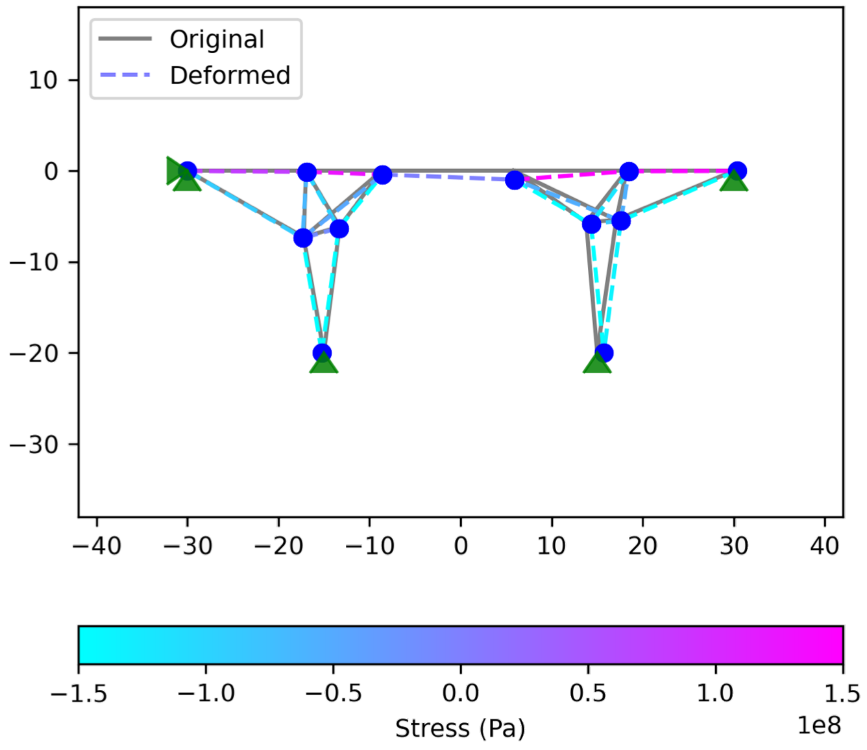


Figure 4.9 Analysis of best achieved bridge with two piers.

CHAPTER 5

CONCLUSION

This research project successfully achieved its main goal, which was to create a computer application with an evolutionary algorithm that could generate and optimize planar truss bridges. After multiple experiments, detailed in Chapter 4, it is possible to verify that the developed algorithm is indeed successful and converges when it tries minimize the structural mass of truss bridges.

A deep study of truss structures, evolutionary algorithms and related literature was done in Chapter 2, which provided the theoretical basis for developing the evolutionary algorithm for this system, as detailed in Chapter 3. Other implementation objectives for this project were also achieved. Multiple fitness functions for the evolutionary algorithm were investigated in Section 3.6, which enabled the possibility for optimizations regarding not only structural mass, but also supported load and structural stress. Evolutionary operations were created for topology, shape and size of the truss structure, as discussed in Section 3.4.

From the beginning, a friendly user interface was set as a goal in order to allow this software to be used by engineers, architects and other professionals designing bridges. Through a user-centric design, as explained in Section 3.8, it was possible to achieve an intuitive and interactive graphical user interface.

When starting the development of this project, it was hypothesized that the generated bridges may showcase known architectural design characteristics, like symmetry, arches and other common truss patterns that result in a better structural stress distribution. While this is not always true, many of the bridges generated in the experiments for this project featured symmetry and known designs. In many cases, arches and variations of common known trusses patterns were observed. In some examples, though, the output bridge had a usual shape. On the other hand, the generated designs were generally very efficient regarding an even distribution of structural stress.

While this program is very successful with respect to its objectives, it is also important to notice that the system has limitations. The structural analysis done by the program is static, which may not be enough for real building projects. Furthermore, the trusses optimized in the program are planar, which can be inadequate when dealing with a three dimensional environment. In these cases, spatial trusses would be required. Also, other variables, such as nodal displacements, are not fully taken into consideration in the fitness functions.

It is possible to conclude that, despite its limitations, the developed system has a lot of potential in aiding the task of designing truss bridges. The system is able to generate planar trusses for any arbitrary set bridge supports, which is a desired characteristic for an engineering tool. The developed application also has the potential to provide insights and help with decision making by showing good performing designs. These optimized designs may be similar to known truss patterns or may be completely new.

5.1 FUTURE WORK

There are many possible directions in which this project could be extended and improved. Regarding the experiments, more of them could be done and at a larger scale. For instance, longer bridges, larger populations sizes and more generations could be employed. This would allow the gathering of more data around the performance of the developed algorithm, for example.

Some of the implemented code was not featured in the experiments shown in this project, such as the sizing mutation and the fitness functions for optimizing structural stress and maximum supported load. Experiments with these fitness functions as well as experiments with mutation of cross-sectional areas of truss members would be definitely enriching to this project.

While the created program focuses on truss bridges, it could also be used for other truss structures with the small additions of classes that inherit from the developed truss abstraction. This gives the program the potential to be useful in the optimization of truss towers, cantilevers and more.

When it comes to accomplishing the goal of using the system to aid the creation of truss bridges, it would be important to do real-world tests with professional bridge designers. This would provide a better understanding of the needs of these professionals and generate a feedback cycle that could improve the application.

Regarding the implementation of the program, the evolutionary operators could be modified for testing different evolutionary approaches. A less elitist selection of parents or a different type of crossover could result in better bridges, for example. The actual implementation of many methods and functions could also deeply benefit from various optimizations and improvements. Many pieces of code take too much processing time and their optimization would mean a faster, more responsive and more useful application.

BIBLIOGRAPHY

[1] Macbook pro (16-inch, 2019) - technical specifications. https://support.apple.com/kb/SP809. Accessed on 04/22/2022. 4

[2] Chadia Abras, Diane Maloney-krichmar, and Jenny Preece. User-centered design. *W. Encyclopedia of Human-Computer Interaction*, 2004. 3.8

[3] Ferdinand P. Beer, Jr. E. Russell Johnston, David F. Mazurek, Phillip J. Cornwell, and Brian P. Self. *Vector Mechanics For Engineers: Statics and Dynamics*. McGraw-Hill Education, 12th edition, 2019. 2.1, 2.1

[4] J. A. Bondy and U. S. R. Murty. *Graph Theory*. Springer, 3rd edition, 2008. 3.1

[5] 2D Boy. World of goo. https://2dboy.com. Accessed on 04/08/2022. 1.1

[6] The Editors of Encyclopaedia Britannica. Truss. In *Encyclopedia Britannica*. Encyclopædia Britannica, Inc., 2014. 1.1, 2.1

[7] Dry Cactus. Poly bridge. http://polybridge.drycactus.com. Accessed on 04/08/2022. 1.1

[8] ClockStone. Bridge constructor. https://clockstone.com/wp/portfolio/bridge-constructor/. Accessed on 04/08/2022. 1.1

[9] Tayfun Dedea, Serkan Bekiroglub, and Yusuf Ayvazc. Weight minimization of trusses with genetic algorithm. *Applied Soft Computing*, 11:2565–2575, 2011. 2.3

[10] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer, 2nd edition, 2015. 2.2, 2.3, 3.7

[11] Timo Eloranta and Erkki Makinen. Timga: A genetic algorithm for drawing undirected graphs. *Divulgaciones Matematicas*, 9(2):155–171, 2001. 3.5

[12] A. Ghali and A. M. Neville. *Structural Analysis: A Unified Classical and Matrix Approach*. Taylor & Francis, 7th edition, 2017. 3.2

[13] R. C. Hibbeler. *Engineering Mechanics: Statics & Dynamics*. Pearson Prentice Hall, 12th edition, 2010. 1.1, 2.1, 2.1

[14] R. C. Hibbeler. *Structural Analysis*. Pearson Prentice Hall, 8th edition, 2012. 2.1, 2.1, 3.2

[15] Kenneth A De Jong. *Evolutionary Computation: A Unified Approach*. The MIT Press, 1st edition, 2006. 2.2

[16] H. Kawamura, H. Ohmori, and N. Kito. Truss topology optimization by a modified genetic algorithm. *Struct Multidisc Optim*, 23:467–472, 2002. 2.3

[17] Kaushik Kumar and J Paulo Davim. *Optimization Using Evolutionary Algorithms and Metaheuristics*. CRC Press Taylor & Francis Group, LLC, 2020. 1.1, 2.2

[18] David J. Lee. *Bridge Bearings and Expansion Joints*. Taylor & Francis, 2nd edition, 1994. 2.1

[19] Qui X. Lieu, Dieu T.T. Do, and Jaehong Lee. An adaptive hybrid evolutionary firefly algorithm for shape and size optimization of truss structures with frequency constraints. *Computers & Structures*, 195:99–112, 2018. 1.1, 2.3

[20] Clemente Marconi. *The Oxford Handbook of Greek and Roman Art and Architecture*. Oxford University Press, 2014. 1.1

[21] MSNBC. Common types of truss bridges. `https://www.msnbc.com/rachel-maddow-show/common-types-truss-bridges-msna199711`. Accessed on 04/08/2022. 2.1

[22] Norapat Noilublao and Sujin Bureerat. Simultaneous topology, shape, and sizing optimisation of plane trusses with adaptive ground finite elements using moeas. *Mathematical Problems in Engineering*, 2013. 1.1, 2.3

[23] Natee Panagant, Nantiwat Pholdee, Sujin Bureerat, Ali Riza Yildiz, and Seyedali Mirjalili. A comparative study of recent multi-objective metaheuristics for solving constrained truss optimisation problems. *Arch Computat Methods Eng 28, 4031–4047*, 2021. 1.1, 2.3

[24] Nantiwat Pholdee and Sujin Bureerat. A comparative study of eighteen self-adaptive metaheuristic algorithms for truss sizing optimisation. *KSCE J Civ Eng*, 22:2982–2993, 2018. 1.1, 2.3

[25] Pedro Jorge De Los Santos. Nusa framework. `https://github.com/JorgeDeLosSantos/nusa`. Accessed on 04/08/2022. 2.1, 3.6

[26] J. Schutte and A. Groenwold. Sizing design of truss structures using particle swarms. *Struct Multidisc Optim*, 25:261–269, 2003. 1.1

[27] Mustafa Sonmez. Artificial bee colony algorithm for optimization of truss structures. *Applied Soft Computing*, 11(2):2406–2418, 2011. 1.1

[28] Pradnya A Vikhar. Evolutionary algorithms: A critical review and its future prospects. *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, pages 261–265, 2016. 2.2

[29] Chun-Yin Wu and Ko-Ying Tseng. Truss structure optimization using adaptive multi-population differential evolution. *Struct Multidisc Optim 42*, 42:575–590, 2010. 1.1

[30] Chun-Yin Wu and Ko-Ying Tseng. Truss structure optimization using adaptive multi-population differential evolution. *Struct Multidisc Optim*, pages 575–590, 2010. 2.3

[31] Teruhiko Yoda and Weiwei Lin. *Bridge Engineering: Classifications, Design Loading, and Analysis Methods*. Elsevier Inc., 2017. 1.1