AIAA SciTech Forum
9 - 13 January 2017, Grapevine, Texas
58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference

10.2514/6.2017-1301

Topological Optimization of a Cuboct Truss Structure Using a Genetic Algorithm

Holly M. Jackson*
NASA Ames Research Center, Moffett Field, California 94035

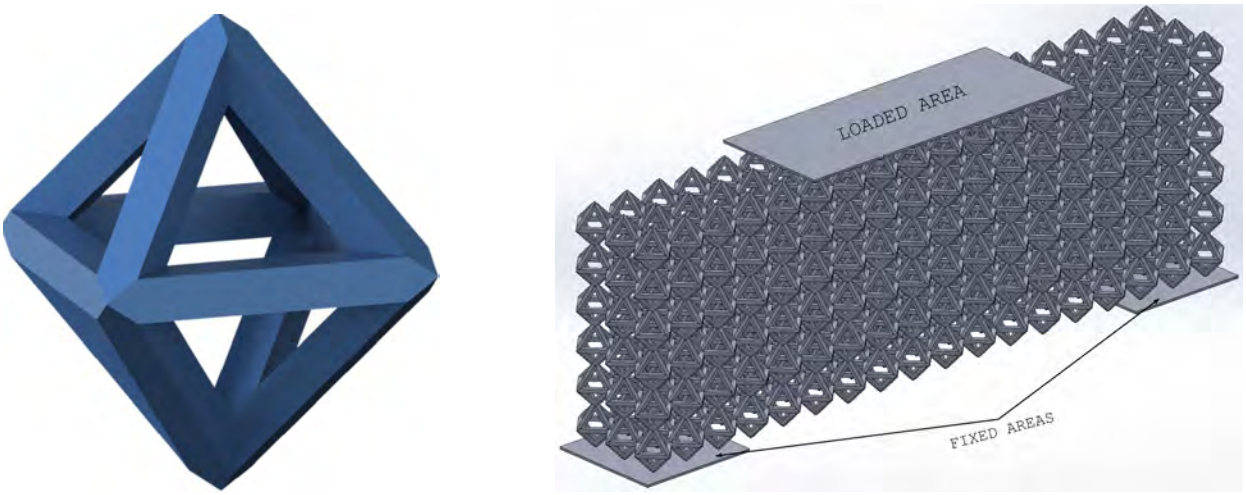
Transporting structures into space poses countless challenges. Cost, size, and weight often heavily constrain the design of structures for space applications. Algorithmic optimization may be particularly applicable for large-scale, long-duration autonomous systems in remote environments, where the exact conditions may not be precisely known ahead of time and/or the conditions could change rapidly. This paper applies a genetic algorithm for the rapid mass optimization of cuboct truss structures with programmable stiffness properties. These low mass, high stiffness structures could significantly reduce the cost required to transport them to space. Four distinct algorithm outputs were 3D printed and verified through stress testing. Performance of the bridge structures was tested by simulation and physical experiment against a conventionally-designed bridge and a full envelope truss beam. Experimental results demonstrate the ability to program mechanical properties and high stiffness-to-mass performance relative to conventional and naive bridges.

I. Introduction

Any structure carried into space must be lightweight due to the high cost of space travel. From habitats on Mars to support structures for the space station, scientists are forced to work with limited material. Every extra kilogram of mass added to a structure increases the total cost of that structure's trip to space by around $10,000.[6] Structures made from lattice framework such as cuboct trusses (see Fig. 1) are actively studied due to their simple modular assembly and high stiffness-to-mass ratio.[4] The improved specifications of these truss structures allow for significant reductions in the expense of space transportation. However, using traditional search methods to optimize a lightweight truss structure would be computationally intractable due to the large solution space (2^n) that could be searched. Creating a versatile, quick, and automated system that would design an optimal structure using resources only where necessary would significantly reduce the cost of transporting structures to space as well as the time required to design and optimize those structures for space applications.

One current method for the topological optimization of lightweight structures is a genetic algorithm, which replicates the process of evolution. Genetic algorithms have been proven to solve optimization problems with complex and competing criteria more effectively than other topological optimization methods such as homogenization.[12] Most conventional genetic algorithms generate an optimal solution from a partitioned solid. The solid structure is divided into n units (such as cubes) of equal size which are removed and added throughout the algorithm to create an optimized structure.[12,13,16] By using a larger existing truss element instead of a solid cube as the base unit in a genetic algorithm, the optimization time and ease of final assembly could be significantly improved. This method of optimization has three major advantages. First, the optimal output would be created from a single repeating truss unit, which opens up many different manufacturing possibilities. Second, this method eliminates the risk of producing bridges that have thin or fragile parts that would be difficult or impossible to construct. Third, since the base truss unit increases the complexity of the lattice, the dimension of the search space, n, could be reduced, which decreases the number of generations, and therefore the time, required to find an optimal solution. This paper describes a new application of a genetic algorithm that utilizes cuboct truss voxels instead of solid cube elements to optimize a lightweight truss structure with programmable stiffness properties.

*Student, *hjackson*18@*ndsj.org*.



(a) Cuboct truss voxel (octahedron)

(b) Full envelope cuboct truss beam with loaded and fixed areas marked (isometric view)

Figure 1: Cuboct truss.

II. Methods

A. Algorithm Outline

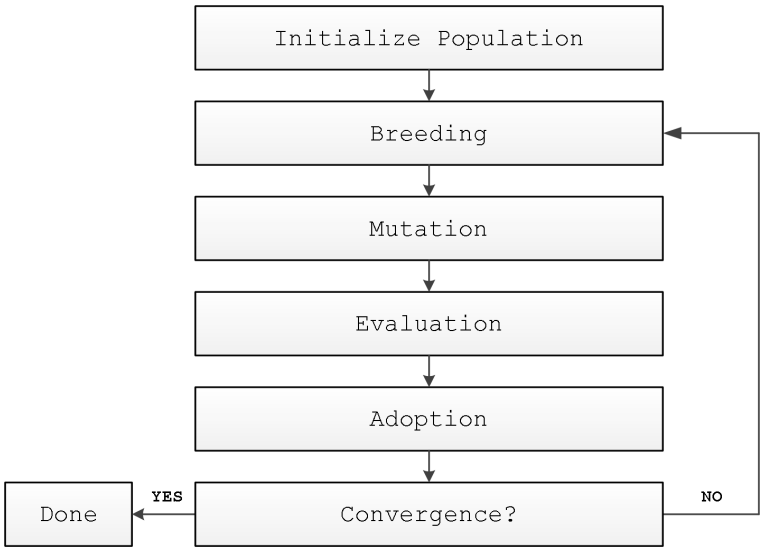


Figure 2: Block diagram of the genetic algorithm.

A genetic algorithm was programmed with conventional object-oriented programming (Java) on a conventional desktop computer. The algorithm replicated the process of natural selection, evolving a population of individual solutions and modifying them every generation. The algorithm had four main steps (see Fig. 2): the creation of the initial population of bridges, the breeding of new population members for the population, the adoption of children (if more fit) into the population, and the repetition of this process over thousands of generations to arrive at a solution.

1. Initial Population

To begin, the algorithm generated an initial population of fifty structures. Each structure was eighteen voxels long, three wide, and six high. A three-dimensional (3D) array containing the state (present or absent) of each voxel was created to represent each structure; this array was the genetic code of the bridge. Up to 5% of the voxels were removed at pseudo-random locations to add variation to the initial population. The displacement of each structure under a 24 netwon (N) load was calculated using *frame3dd*, a direct-stiffness iterative solver for the structural analysis of trusses and support structures.[8] To ensure the viability of the initial population, bridges exceeding yield strength under load were discarded and replaced by new, viable structures.

2. Breeding/Mutation

To produce children, bridges in the initial population were randomly paired as parents. The 3D structure arrays of the parents were merged (50/50 selection from each parent) to produce a child. Random mutations were then added to the new childs structure array. The added mutation rate alternated between 2% and 20% each generation of children. The most similar parent who had given the most genes, or voxels, to the child was recorded for later use.

3. Evaluation/Adoption

Frame3dd was called to calculate the predicted displacement of each child bridge under 24 N. The displacement values of the parent bridges were retrieved from stored arrays. Each child and its parents were assigned a corresponding fitness score. A bridge's fitness score was defined as the weighted sum of its displacement under 24 N and its mass, which was quantified as the number of voxels present in the structure. In some runs of the algorithm, the displacement weighting was scaled, so it played a larger role in adoption. In other runs, a displacement cap was added to discard unfit children who did not meet a minimum stiffness. The algorithm allows the user to change the value of stiffness or mass during adoption prior to algorithm execution.

If the child had a lower fitness score than its most similar parent, the child replaced that parent in the population. However, if the child had a higher (worse) fitness score than its most similar parent, the child was discarded, and both parents retained their place in the population.

4. Iteration

After all children and parent fitness scores had been evaluated and compared, a new generation began with the updated parent population. The above process of breeding and adoption was repeated until all the fitness scores of the population members converged, and one optimal bridge was created. This reproduction took approximately 7800 generations with a typical execution time of seventy-two hours on a 12-core computer. Typically one optimal solution was found. However, in two out of the four runs executed in this experiment, a single outlier (out of fifty population members) never converged.

B. Determining the Mutation Rate

As described in II.A.2, the genetic algorithm added random mutations to the genetic code of the population each generation to create variation. A wide variety of advice exists in literature surrounding the optimal mutation rate to use in genetic algorithms. Some sources suggest that high mutation rates are optimal because large variations can cause rapid initial convergence and avoid local minima.[10] Other sources promote low mutation rates because, as the solution nears optimum, smaller refinements are needed to converge.[19] Even still, other sources say that the optimal mutation rate varies depending on the application.[17,20] To discern the optimal mutation rate for the population, simple tests were run using smaller scale bridges. Low mutation rates, high mutation rates, and a combination of both were tested. It was found an alternating combination of a high mutation rate (20%) and a low mutation rate (2%) produced fast, early convergence and high-quality end optimization.

C. 3D Printing

The final optimized bridge design was selected from each algorithm run. The algorithm was coded to output each optimal bridge in SB file format that could be opened up in Antimony, a CAD tool developed by Matthew Keeter at MIT.[14] The result was exported from Antimony as an STL file, and that STL file was imported into MeshLab, an open-source tool for editing 3D meshes.[2] The STL file was compressed using *Quadric Edge Collapse Decimation* in MeshLab to speed printing.

Three-dimensional solid models were then printed from the STL files. Five copies of each genetically optimized bridge, five copies of a conventionally-designed (standard) bridge, and five copies of a full envelope truss beam (no missing octahedrons) as a control were created on a combination of FDM and SLA printers.[9] The bridges printed on the FDM printer were ABS plastic and 100% scale (21 x 4 x 7 cm). The bridges printed on the SLA printer were acrylic plastic and 50% scale.

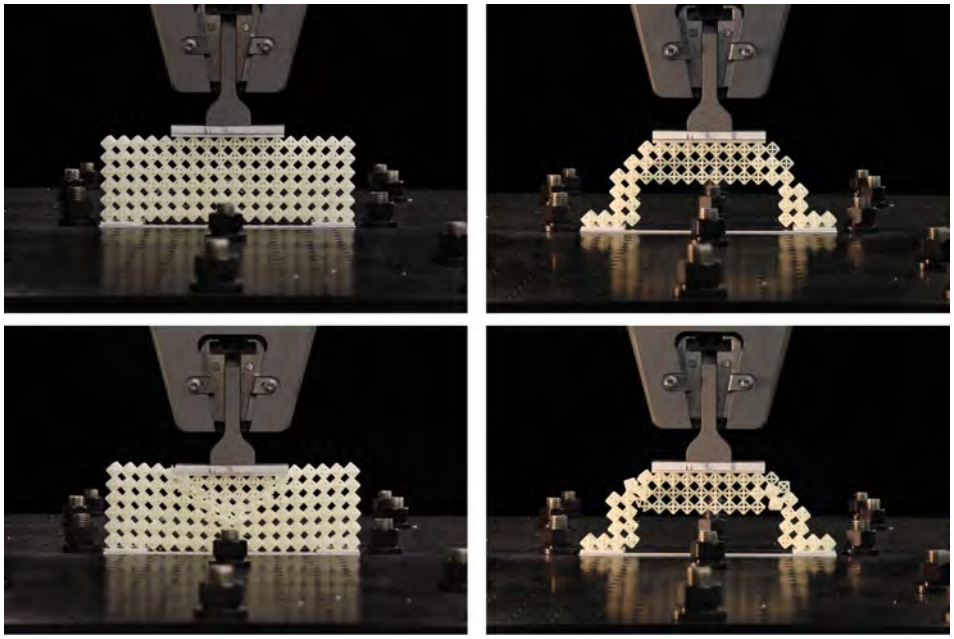


Figure 3: 3D-printed bridges with delrin fixtures in Instron load testing apparatus (shown before and after testing).

D. Stress Testing

The algorithm-generated bridges, the standard bridge, and the full envelope were all stress tested using an Instron 5982 load testing system.[1] Fixture plates were created using a laser cutter and applied to the bridges using epoxy to hold them in place during stress testing (see Fig. 1b and Fig. 3 for placement). A constantly increasing load was applied to the 3D-printed bridges until they failed. Displacement (in mm) was recorded at increments of 0.5 N of pressure. In particular the displacement at 24 N, which had been predicted using *frame3dd*, was recorded.

III. Results

The genetic algorithm was run four times with varying parameters and produced four different optimal bridges. In the first two runs, high-mass structures were penalized; in these runs, the mass played the largest role in the final fitness score. As a result, the two bridges created were both low in mass but showed increased displacement owed to the low emphasis of stiffness in the fitness calculation. As can be seen in Fig. 5, the bridge output from the first run of the algorithm was 45% the mass of the standard bridge, but displaced 492% more under load.

In the third and fourth algorithm runs, a much larger emphasis was placed on the displacement in the final fitness scores. The displacement cost parameter in the algorithm was scaled up by a factor of 10. In the

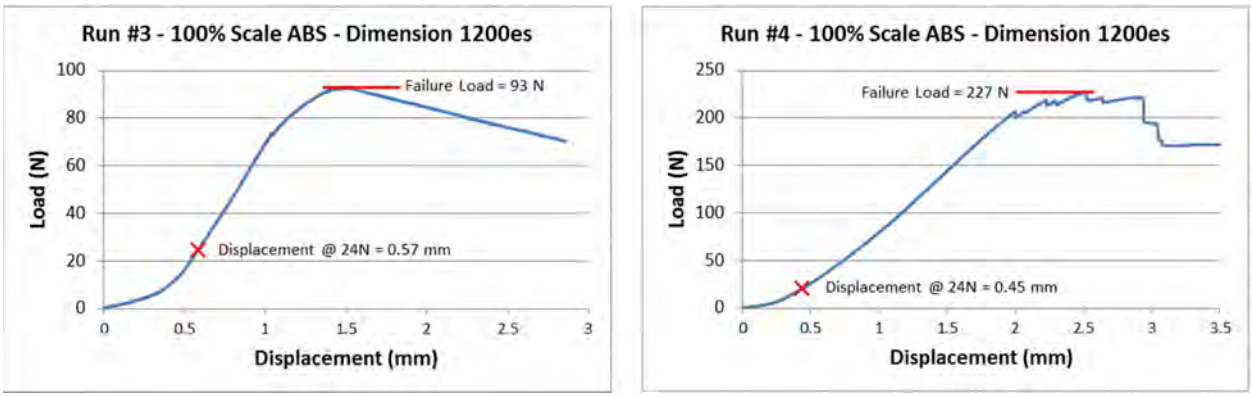
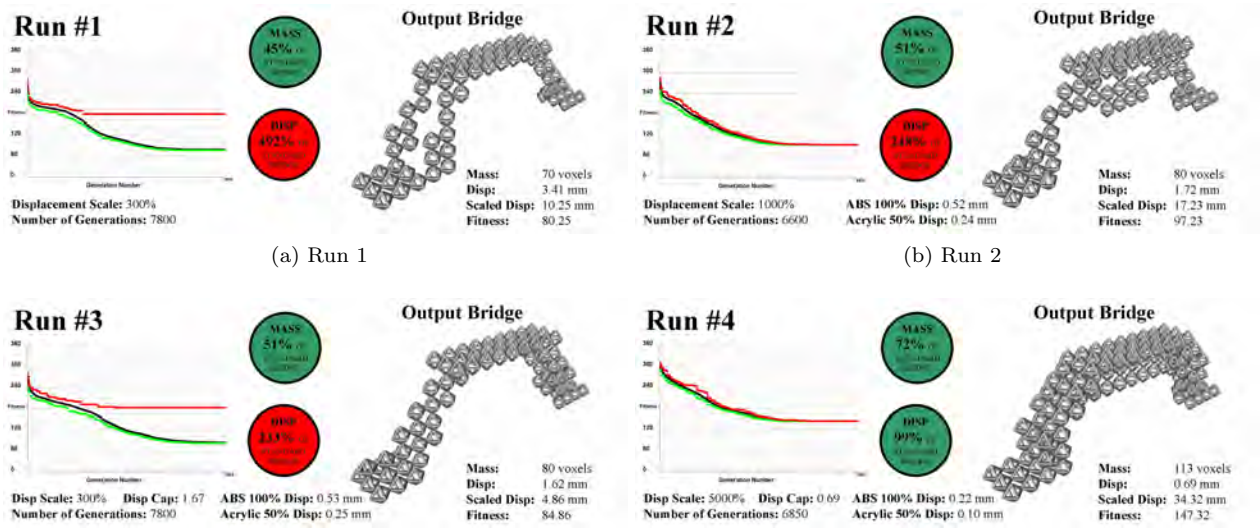


Figure 4: Load versus displacement plot of the stress tests for two 3D-printed bridge samples with load at failure and displacement at 24 N marked.



(a) Run 1

(b) Run 2

(c) Run 3

(d) Run 4

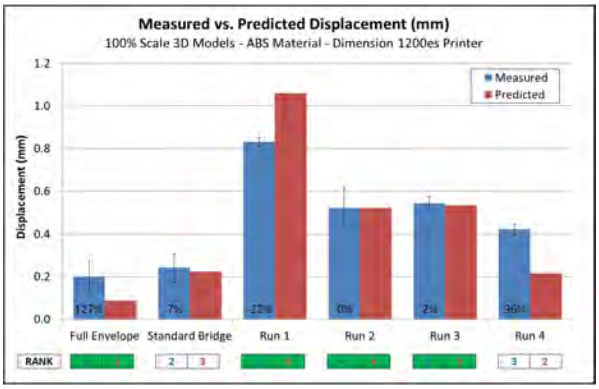
Figure 5: Descriptions of all four bridges optimized by the genetic algorithm and their corresponding optimization parameters.

fourth run, a restrictive displacement threshold was also added: the population was pruned every iteration to those that displayed less than or equal to the displacement of the standard bridge. In run four, the algorithm converged upon a design with 72% the mass of the standard bridge and equal (99%) displacement.

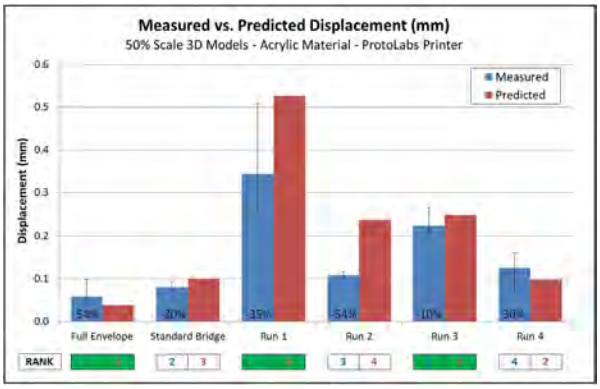
IV. Discussion

Fig. 6a displays the predicted displacements for the bridges printed on the FDM printer at 100% scale in millimeters as well as the measured displacement values from real 3D tests. The average displacement value and error bars measured from the two samples for each bridge type are graphed. In addition, below each chart, the relative predicted and measured stiffness for comparison have been recorded. Overall, four out of six relative stiffness rankings had been predicted correctly. The two incorrect rankings were a swap between run four and the standard bridge. These two bridges were predicted to have the same displacement, so this switch in rank is not surprising.

Fig. 6b shows the predicted and measured displacements for the bridges printed on the SLA printer at 50% scale. There were three SLA samples for each bridge type. For these bridges, three out of six of the



(a) Plot of the measured and predicted displacements (in mm) for each bridge type printed at 100% scale (21 x 4 x 7 cm) on an FDM 3D printer using ABS plastic. The chart underneath the plot shows a comparison of the relative stiffness predictions and measurements.



(b) Plot of the measured and predicted displacements (in mm) for each bridge type printed at 50% scale (10.5 x 2 x 3.5 cm) on an SLA 3D printer using acrylic plastic. The chart underneath the plot shows a comparison of the relative stiffness predictions and measurements.

Figure 6: Measured and predicted stiffness comparison plots for the two 3D printed bridge types (FDM and SLA).

relative stiffness rankings were predicted correctly. Overall, less accurate results were achieved using SLA. This is attributed to a manufacturing error observed with the SLA fabrication where certain truss members were consistently smaller in dimension than the original design.

Overall, the difference between the measured and predicted displacement values ranged from 0% to 127%. Taking measurement error into account, the algorithm accurately predicted the displacement in half of the cases.

V. Conclusion

Current genetic algorithms typically evolve structures from a starting solid partitioned into n equally-sized cubes. The genetic algorithm described in this paper differs from other genetic algorithms because it uses an existing truss element, the cuboct truss, as its base element. Since the genetic algorithm starts with a stiff, lightweight, and modular building block (the cuboct truss), the final structure is simple to assemble and the computational complexity of the algorithm is reduced.

Another benefit of this algorithm is its applications to space structures. This algorithm is a quick, efficient, and useful tool that can reliably create lightweight cuboct truss structures with programmable stiffness properties. This paper has demonstrated its effectiveness at optimizing stiffness and mass of bridge structures. In the future, it could be adapted for other structures and modified to optimize additional parameters such as strength, cost, material type, and reusability. This algorithm brings particular value for space applications by minimizing the exorbitant expense to loft heavy structures into space.

Acknowledgments

The author gratefully acknowledges the valuable advice and support of Dr. Kenneth Cheung and Daniel Cellucci. Essential materials and test equipment for this work were made available by the Coded Structures Lab at NASA Ames Research Center.

References

[1]5980 series dual column floor frames manual. 2009.

[2]Meshlab. 2014.

[3]Ishanu Chattopadhyay and Hod Lipson. Abductive learning of quantized stochastic processes with probabilistic finite automata. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 371(1984):20110543, 2013.

[4]Kenneth C. Cheung and Neil Gershenfeld. Reversibly assembled cellular composite materials. *Science*, 341(6151):1219–1221, 2013.

[5]Kim Christensen. Percolation theory. *Imperial College London, London*, page 40, 2002.

[6]Nancy Cicco and Christine Gillete. Nasa funds development of new structures for space travel. 2015.

[7]V.S. Deshpande, M.F. Ashby, and N.A. Fleck. Foam topology: bending versus stretching dominated architectures. *Acta Materialia*, 49(6):1035–1040, 2001.

[8]Henri P Gavin. Frame3dd user manual. 2010.

[9]Christian Groth, Neal D. Kravitz, Perry E. Jones, John W. Graham, and W. Ronald Redmond. Three-dimensional printing technology. *J Clin Orthod*, 48(8):475–485, 2014.

[10]Randy L. Haupt. Optimum population size and mutation rate for a simple real genetic algorithm that optimizes array factors. In *Antennas and Propagation Society International Symposium, 2000. IEEE*, volume 2, pages 1034–1037. IEEE, 2000.

[11]Jonathan Hiller and Hod Lipson. Dynamic simulation of soft multimaterial 3d-printed objects. *Soft Robotics*, 1(1):88–101, 2014.

[12]Jonathan D. Hiller and Hod Lipson. Multi material topological optimization of structures and mechanisms. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1521–1528. ACM, 2009.

[13]Max Hultman. Weight optimization of steel trusses by a genetic algorithm. *Department of Structural Engineering. Lund University*, 2010.

[14]Matthew Keeter. Antimony.

[15]Ying-Hong Liao and Chuen-Tsai Sun. An educational genetic algorithms learning tool. *IEEE Trans. Education*, 44(2):20, 2001.

[16]David Liu, David Walker, and Alan Jennings. Topology optimization of an aircraft wing. 2015.

[17]K. Matthias, Thomas Severin, and Horst Salzwedel. Variable mutation rate at genetic algorithms: Introduction of chromosome fitness in connection with multi-chromosome representation. *International Journal of Computer Applications*, 72(17), 2013.

[18]Marc André Meyers, Joanna McKittrick, and Po-Yu Chen. Structural biological materials: critical mechanics-materials connections. *science*, 339(6121):773–779, 2013.

[19]Marek Obitko. Introduction to genetic algorithms: Recommendations.

[20]Patil V.P. and Pawar D.D. The optimal crossover or mutation rates in genetic algorithm: A review. *International Journal of Applied Engineering and Technology*, 5(3):38–41, 2015.