Generation of Optimal 2D Truss Structures Via a MASTAN-Integrated Genetic Algorithm in Matlab

Stanford University

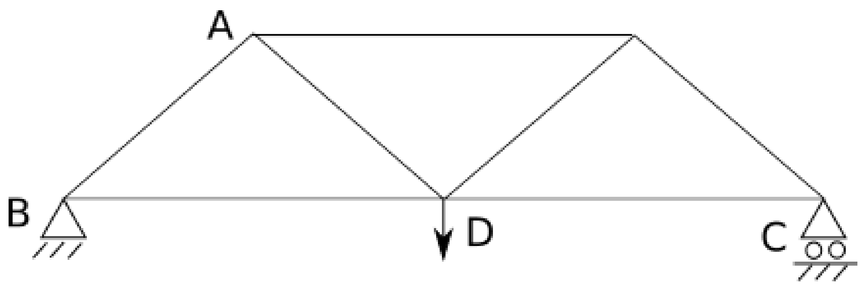
Computational Micromechanics of Materials Lab

R. Cohen (rrcohen@stanford.edu)

Mentors: E. Lejeune and R. Gibbons; Advisor: C. Linder

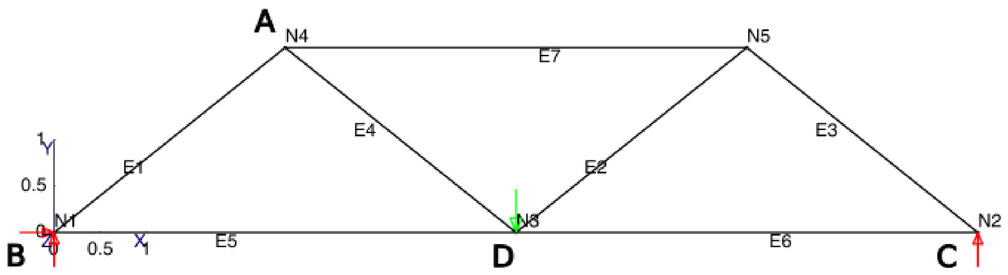
Stanford | ENGINEERING
Civil & Environmental Engineering

Introduction to Trusses



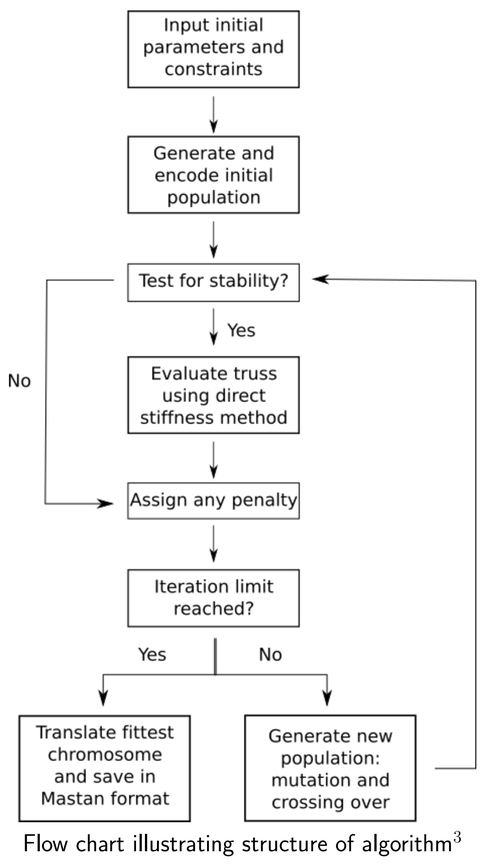
Basic 2D Truss diagram. A) Members connect at pivoting joints. B) Support condition fixing X and Y degrees of freedom at node. C) Support condition fixing Y degree of freedom at node. D) Applied load condition at node.

- A *truss* is a structure of axial members (elements) connected at pinned joints (nodes), such that no moment may be transferred between members.[1]
- Trusses are designed to efficiently distribute a load.[1] Examples of truss structures range from bridges and cell phone towers to internal and external supporting components in additively manufactured devices.
- One method of efficient truss design that is particularly relevant for additive manufacturing applications (where engineering complexity is not an issue) is weight minimization.[2]
- Matlab-based program *MASTAN2* is used to visualize, analyze, and evaluate truss structures.



Computer representation of the truss structure above, represented in MASTAN's visualization tool and stored in MATLAB matrices. To generate this truss, the user defines node location, element connectivity, member material properties, applied load, and fixed supports. The red arrows indicate displacement fixed supports while the green arrow indicates applied load.

Genetic Algorithm



Flow chart illustrating structure of algorithm[3]

Goal: Implement a genetic algorithm to automatically generate and optimize 2D trusses to support a given load in a specified design space under prescribed support conditions and material parameters.

- A *Genetic Algorithm* imitates natural selection by producing iterative generations of increasingly fit populations until converging on a near optimal solution.[3]
- The algorithm evaluates the fitness of each member of a population of possible solutions and "breeds" the most-fit members to generate the subsequent population.[3]
- Each truss is encoded in a binary "chromosome"[3], a series of binary digits representing the connected members of the truss.
- The breeding step includes crossing over (cutting and recombining two fit trusses to blend their attributes) and mutation (random switching of binary values during the breeding step to introduce new possible solutions and maintain diversity).[3]

Truss Fitness Evaluation

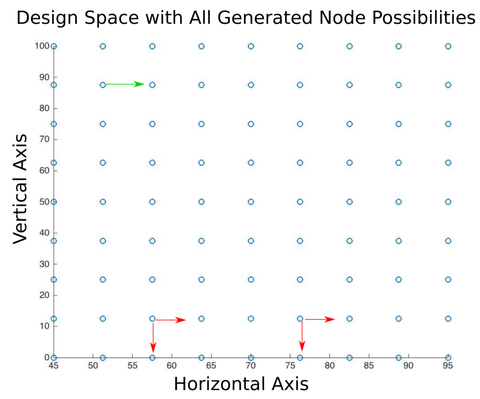
- Fitness value represents how well a truss meets the design goal (minimized material). Lower fitness values are optimal.
- Minimize fitness f defined as $f = \sum_i V_i + \phi_g + \phi_i$ where V_i is the volume of member i and ϕ_g is a global penalty function and ϕ_i is a local penalty function.
- Penalties worsen the fitness values of trusses that exceed specified restrictions on nodal deflections and internal element forces.[3]
- Nodal deflection and internal element forces calculated using *direct stiffness method*, implemented specifically for this algorithm:
  - Element stiffness equations are matrix equations that relate the element force vector $\{F\}$ and displacement vector $\{\Delta\}$: $\{F\} = [k]\{\Delta\}$[4]
  - Element stiffness matrix $[k]$, analogous to the Hooke's Law spring constant, is constructed from the prescribed area A, length L and Young's modulus E of the member.[4]
  - After constructing the stiffness matrix for each element of the truss, the *global stiffness equation* is constructed to include the displacements and forces of each member. The global stiffness matrix combines corresponding terms from all element stiffness matrices.[4]
  - The global equation matrices are partioned by the degrees of freedom involved: free (f) or supported (s).[4]
  - $\{\Delta_f\} = [k_{ff}]^{-1}\{P_f\}$ is solved to obtain nodal displacements.[4]
  - Given the solved nodal displacements, the element stiffness equations $\{F^i\} = [k^i]\{\Delta^i\}$ are solved to obtain the internal force distribution for each element i.[4]

Key Features of Genetic Algorithm

MASTAN2 Integration User defines geometry of problem in MASTAN's graphical interface. Four outermost nodes define the available design space, and nodes with support and load conditions establish the optimization problem.

MATLAB Interface Algorithm uses text-based interface to allow user to input key parameters, including number of generations, population size, and various parameters defining random truss generation and mutation.

Grid of Possible Nodes The algorithm fills the design space (see right) with a grid of possible node locations. The final truss solution consists of elements connecting some subset of these nodes.



Represenation of design space generated by algorithm. Blue circles represent all possible nodes between which elements may be placed, red arrows represent support conditions, and the green arrow represents the load.

Initial Population Generation The algorithm randomly generates an intitial population of trusses and ensures complete interconnectivity of all essential nodes (support and load locations). It then randomly adds and removes elements based on a mutation probability.

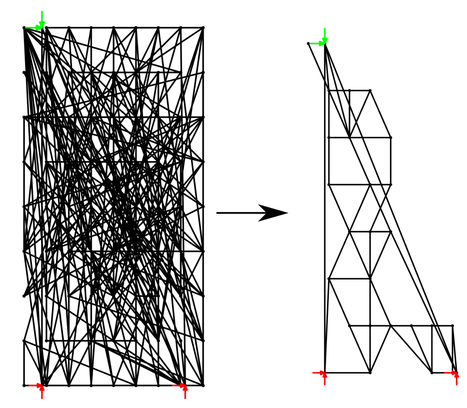
Unique Truss Encoding Trusses are stored in a binary chromosome used in the breeding step, and are converted to and from matrices for fitness evaluation. Each spot in the chromosome represents a pairing between two possible nodes in the design space. A "1" represents an element connecting the two nodes. The pairings progress in sequential order:

1	0	0	1	1	0	0	0	1	0
1-2	1-3	1-4	1-5	2-3	2-4	2-5	3-4	3-5	4-5

The binary chromosome above represents an encoding of a truss with 5 possible nodes in the design field. The numbers in red signify the nodes to be connected by an element if the slot immediately above is "1".

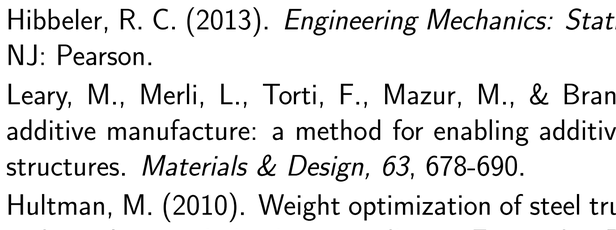
Elitist Selection The top ten trusses from each generation are passed identically (unmutated) to the subsequent generation, guaranteeing that the top fitness will not deteriorate from one generation to the next.[3]

Element Removal Preference In order to encourage the program to find lighter (more fit) solutions, the algorithm is 60 times more likely during the mutation step to randomly remove an element than to randomly add one. This significantly increases the rate of fitness improvement and prevents premature convergence.



Transformation from randomly generated initial truss (left) to final result (right).
Population size: 2000. Number of iterations: 1000

Results



The fitness plot shows that convergence is obtained after approximately 750 runs.

- While the fitness certainly moves toward a more optimal value, even visual inspection of the resulting truss diagrams suggests that the progam does not find an absolute optimum.

Summary and Acknowledgements

- The algorithm effectively converges towards a more optimal truss design.
- A more strongly guided method for generating the initial population may allow for an improved final result.
- Thank you to Emma Lejeune and Ren Gibbons for mentorship, Christian Linder for sharing his lab and resources, Berkin Dortdivanlioglu for assistance in using the remote workstations, and Stanford CEE VPUE for funding this project.

References

1. Hibbeler, R. C. (2013). *Engineering Mechanics: Statics and Dynamics*. Upper Saddle River, NJ: Pearson.
2. Leary, M., Merli, L., Torti, F., Mazur, M., & Brandt, M. (2014). Optimal topology for additive manufacture: a method for enabling additive manufacture of support-free optimal structures. *Materials & Design, 63*, 678-690.
3. Hultman, M. (2010). Weight optimization of steel trusses by a genetic algorithm-size, shape and topology optimization according to Eurocode. *Rapport TVBK-5176*.
4. McGuire, W., Gallagher, R. H., & Ziemian, R. D. (2000). *Matrix Structural Analysis, 2nd Edition*.