Topology and Weight Optimization of a 3D Truss by Numerical Method

Arash Mohammadzadeh Gonabadi 1,a, Mohsen Mohebbi 1,b, Ali Sohan Ajini 1,c

1 Department of Mechanical Engineering, Islamic Azad University, Parand Branch, Tehran, Iran
a arash_mg@semnan.ac.ir
b Mohsen_mohhebbi@ymail.com
c alisohanajini@gmail.com

ABSTRACT. One of the difficulties in designing a truss is finding the best condition for topology with the minimum weight. Today, several techniques are presented for searching in computer science to find an approximate solution for optimization and research problems that counts as a subset of artificial intelligence. Genetic algorithm is a special type of evolution algorithms, which uses evolutionary biology techniques such as inheritance and mutation. In fact, genetic algorithms use Darwin's natural selection principles to find optimum formula to forecast or fit the pattern. Genetic algorithms are often good options for forecasting techniques based on accidents. Briefly, it is said that algorithmic is a programming technique which uses genetic evolution as a solution pattern. Using this method along utilizing Matlab and Ansys, we can design a 3D truss by having constraints, supports and applied forces on truss in the optimum condition (geometry and weight) and by connecting these two software we can find FEM results, as well.

Introduction. With the development of science and technology and by relinquishing old methods, which were in form of try and error and in addition to taking too much cost, they take too much time, before conjugating too much money and time, by converting the problem into mathematical model and doing some simple setting, we can find results close to reality.

In the modern age, nobody is looking for old methods anymore and all majors are looking for different software for speeding up the process of finding the results. Today, various optimization methods have made it easy to find the best answers. Among optimization methods, we can point to genetic algorithm, particle swarm algorithm and competitive colonization and so on. Today, genetic algorithm has found its own place among other optimization methods for solving complex problems and counts as an effective and efficient method for solving such problems in business, scientific and engineering fields.

Working range of genetic algorithm is so wide and with daily increasing development of science and technology the application of this method has been widely developed in optimization and problem solving. Genetic algorithm is one of the subsets evolutionary calculations that has a direct relationship with artificial intelligence and it is, in fact, one of the subsets of artificial intelligence. Genetic algorithm can be a general searching method that imitates biologic evolution rules. Genetic algorithm applies the survival of the best rule on a series of problem's answers to find better answers. In each generation, using selection process suitable to answer values and reproduction, the selected answers find better approximation for final answer using activisms imitated from natural genetic. This process makes it possible for new generations to be more adaptive to problem conditions.

Now it is enough to model mentioned problem mathematically. By equating the parameters of the problem with genetic algorithm parameters and applying constraints, which directs genetic algorithm to create reasonable and correct answers, we can find the best and most optimum answer for the intended problem. Genetic algorithm is emanated from Darwin's evolution process. According to
Darwin's evolution process, only people with desirable genes can survive and reproduce new children, thus people with shoddy genes will be eliminated.

Rechenberg mentioned the main idea of evolutionary algorithms for the first time [1]. In fact, genetic algorithms that are subsets of these algorithms are computerized search methods based on optimization algorithm, genes and chromosomes geometry that was mentioned by Prof.Holland in Michigan University and was developed by a group of his students later [2].

For optimizing a truss using genetic algorithm there are four steps as follows:

1. Optimizing the section, which includes determining the optimum section for members [3, 4, 50].
2. Optimizing the shape or geometry of which the optimum constraint coordination for constant topology (constant connectivity) of members are determined [6, 50].
3. Topology optimization, which contains determining the best layout of members in, assumed space of constraint coordination [7, 8, 50].
4. Geometry optimization, which includes three previous steps together at the same time.

In this article, using genetic algorithm methods, a new method is represented for topology and cross section optimization of truss constructions. Classic methods, for this purpose, require special mathematics, but this end is easily accessible through genetic algorithm [9]. People like Grisson and partners; Rajan [10, 11], Hajla and partners; Rajio and Chrishnamourti [13]; Ozaki [12]; Chai and partners [15]; Vasques [14, 18, 17]; Kave and partners [16]; Sisok [20]; Tang and partners [19, 22]; Rahami and partners [21], D and partners [24]; Cheng and partners [23] and so on have proven this matter using their suggested methods (GA) for topology, cross section, and truss constructions optimization. Additionally, Arash M. Gonabadi [50] has optimized the topology and weight of a truss using Imperialist Competitive Algorithm (ICA).

Now, by combining the modelling and one of optimization algorithms, we can reach a reasonable and optimum answer.

**Truss optimization formulation.** The objective function is the one we want to optimize using genetic algorithm. Genetic algorithm has approximately a general condition and is not only mentioned for a specific problem but it has provided the possibility that by applying constraints, it will be specified for mentioned problem [35-42, 50].

**Constraint.** Constraint literally means promise and covenant. In engineering applications, it means that the mentioned system or action is supposed to be adhering to a series of rules and these rules are called constraint. For example, assume a simple mechanism that consists of several joints and supports, which create a regular and predictable movement. Hence, the mechanism has a series of constraints (joints and supports) that is always depending on them while moving [45-51].

In optimization algorithms, due to different applications, we must define constraints depending on the mentioned problem while using the algorithm. The defined constraints speed up the optimization problem and sometimes prevent error creation while the program is operating. In mentioned algorithm, we can apply constraints using these two methods such as: penalty function and using constraint functions in algorithm geometry [50].

**Penalty Function.** In this method, we import the constraint in main function and involve it in the answer. In optimization algorithms, some inputs are produced and then are used in fitness function (the function that is to be optimized) then an output is created. Optimizing the output function is our purpose. If the output of fitness function is not a single number, and consists of two or more numbers, there is a function that is representing a cantilever beam that has a force applied on the other end [35-40].
This example can have several outputs: stress, strain, bending and so on. For this example, at the end of fitness function, we sum up all the outputs that we want to optimize and model them as a number. Then we set constraints in the fitness function with Penalty function. Knowing whether this constraint is established or not, so the output of constraint is either 0 or 1 meaning that zero is correct and one is incorrect. Thus, if the constraint is correct (meaning that the output of constraint is zero) when it is multiplied in penalty magnitude, it will make the value of penalty zero. Now, if there are several constraints, the penalty magnitude will be multiplied in every output of every constraint, and will result in increasing the final value of fitness function [40-51].

\[ F(X) = f(X) + \sum rp \times P(X) \]  

where \( F(X) \) – the output of fitness function with applied constraints;  
\( f(X) \) – the output of fitness function;  
\( rp \) – penalty;  
\( P(X) \) – equity or inequity answer;

**Using constraint function in algorithm construction.**

In genetic algorithm, we can define the constraints independently and the algorithm will select the inputs for fitness function according to these constraints [35-42, 51].

**G1 Constraint.** In every truss construction, there is a series of important nodes, which must exist in the construction for every reasonable design. These important nodes are called main points. These nodes are the ones that a force is applied on or are on a support [35-42].
Fig. 2. Main nodes of truss.

Nodes 1, 2 and 3 are main nodes.

**G2 Constraint.** The kinematic steadiness is analyzed in this constraint meaning the construction must not be a mechanism.

According to degree of freedom (DOF) from Grobler criteria, we have:

\[ D = m + r - 2j \]  

(2)

where \( m \) – number of links \((i - n)\) every step of algorithm;

\( r \) – Number of supporting reactions;

\( j \) – Number of truss constraints;

If \( D \leq 0 \), the construction is not a mechanism and has a static state.

**The overlap constraint:** We must be careful to check that a link is not overlapping another one in a construction, for example, in Fig. 3 there are two links between nodes 1, 2 and 3 and there is one link between node 1 and 3 that must be eliminated.

Fig. 3. Overlapping links.

**Single-linkage constraint:** If a point is just used once in a topology matrix and is not a main point, the link connected to that point will be eliminated.
Fig. 4. The link between nodes 3 and 4 must be eliminated.

Duplicate linkage constraint: the construction is analyzed to eliminate the linkage that is repeated in topology matrix.

Fig. 5. Repeated link.

Empty linkage constraint: the construction is analyzed to eliminate the linkage that is empty (representing only one point) in topology matrix.

Fig. 6. Empty link.

After applying initial constraints, we are going to apply secondary constraints related to the outputs of fitness function or mentioned answers which are as follows [35-45, 51]:

- The stress of every link must not exceed allowable stress amount
- The deflection magnitude of nodes must not exceed allowable amount
- The cross section of links must be in allowable limitation

Applying topology optimization steps:
Choosing an initial topology.

Analyzing the initial constraints (In case of dissatisfaction go back to first step, in case of satisfaction go to the next step).

Analyzing the construction to find stress and dislocation magnitudes

Are steadiness, stress and dislocation constraints satisfied? (In case of dissatisfaction go back to first step, in case of satisfaction go to the next step).

Applying genetic algorithm method and gaining a new topology.

Repeating this process until stoppage point.

Size optimization

As this is a two-step method, after gaining the optimum topology in second step, we begin optimizing the size. The existence or inexistence of a link is clear in this step and we only optimize the cross sections of existing links in construction. The cross section of links are chosen continuously [40-45]. The objective function is the weight of construction and can be calculated from this relation:

\[ m = f(x) = \sum_{j=1}^{n} p_j \times L_j \times A_j \quad (3) \]

Assuming a limitation for the cross section of links which is:

\[ A_{\text{min}} \leq A \leq A_{\text{max}} \quad (4) \]

Thus, in size optimization, the design variables are cross sections of links that are continuously chosen from \( A_{\text{min}} \leq A \leq A_{\text{max}} \). We must choose arrays for every particle as long as design parameters (the existing links in second step) meaning that each particle must be considered as a vector of which the arrays are as long as design parameters:

\[ X_i = [A_{i1}, 1, A_{i2}, 2, \ldots, A_{in}] \quad (5) \]

The implementation plan

The goal of this section is creating the matrix model of the truss [40-50].

Defining the truss in writing code

To define these type of constructions, we must define the location of points that show the beginning and end of links, at first. The number of columns shows the magnitudes of X, Y, and Z points and the number of rows shows the number of points that create the construction. In other words, if the construction has two columns it is a 2D construction and if it has three columns, it is a 3D one. In Fig. 7 the points are near each other with a 1000-Inches length between each of them.
How to define links between points

In this project we use this method of which a \( (n \times D) \) array is created and \( n \) represents the number of links in constructions. On the other hand, every row contains information about every link.

In this method, the first column contains the number of point that a link starts and the second column contains the end point of the link. Using this method to show the truss can keep a lot of information in itself. For example, we can add a third column to matrix that can keep the cross section area of every link in itself.

Fig. 7. Shows how to define truss coordinates.

Fig. 8. The number of truss elements.
For this matrix, we can assume a standard of which in every row, the array of the first column must be smaller than the array of the second one, and all the arrays of the first column must be arranged from the most to the least. This will increase the search speed. Hence, we can rewrite the above matrix like this:

\[
\begin{pmatrix}
1 & 2 & 0.4 \\
1 & 4 & 0.2 \\
2 & 3 & 0.4 \\
2 & 4 & 0.05 \\
3 & 4 & 0.3 \\
\end{pmatrix}
\]  

\(\text{Number of nodes} = (7)\)

**Importing forces in coding**

We import the forces in form of the matrix shown below which explains that the force is applied in Y-direction in node 2 and node 3.

\[
\begin{pmatrix}
0 \\
0 \\
0 -100 \\
0 -100 \\
0 \\
\end{pmatrix}
\]

Connectivity table = \(\text{(9)}\)

**Importing supports in coding**

We can define supports as it is explained in section (2-4), the difference is that if there is a support in a node, we use 1 in the directions that the support is preventing movement.

**Defining an object function for a truss**

Fitness function is the one that is to be optimized. Our goal is to find a mathematical model for a 3D truss that provides us the ability of analyzing the truss in Ansys software. This function must act in such a way that by receiving inputs, the function will process all the topology creation of the truss (truss geometry) including supports, forces and links. Then, we import the resulting truss in Ansys, and the software will report the desired outputs of the problem (stress, strain and bending) as the output.

**Truss in Ansys**

Among designing steps of a truss in Ansys we can point to: defining the types of elements, the material of the truss, nodes, the elements between nodes, supports, forces and entering strength modules. Among Ansys abilities, we can point to Session Editor [43-51] (see Fig. 9).
This command shows all the steps that are done in Ansys, on the other hand, it shows all the done activities in design process. For example, a simple truss is designed in the Ansys (Fig. 10).

Fig. 9. Session editor.

Fig. 10. Designing a simple truss in Ansys.

Now we choose session editor command, see Fig. 11 and Fig. 12.
Now if we run this text file in command section of Ansys, this will create the truss again. If we create the lines of this file using Matlab software, we can create a different truss by entering intended inputs (elements and cross sections) [37-54].
The solved example of 3D Truss

Problem Description

There is a 3D truss with 12 nodes, 4 supports, and 2 forces. For optimizing the truss geometry and cross sections using topology genetic algorithm we need material properties of links.

Elasticity module: $2 \times 10^9 \text{[pa]}$

Poisson’s ratio: 0.3

Connectivity table =

$$
\begin{bmatrix}
0 & 0 \\
0 & -100 \\
0 & -100 \\
0 & 0 \\
0 & 0
\end{bmatrix}
$$

Support=

$$
\begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix}
$$

Solving

We should solve this problem by Ansys and Matlab [52, 53], see:

Fig. 13. Topology optimization.
Fig. 14. Topology optimization.

Fig. 15. Penalty Function-Steps of Optimisation, topology optimization process, Matlab, history report.

Fig. 16. Penalty Function-Steps of Optimisation, cross section optimization process, Matlab, history report.
Discussion and conclusion

In this text, a new optimization algorithm based on mathematical modelling of genetic process for truss optimization was presented. Different methods were explained for optimization. Some of these methods repeatedly and based on gradient, find the optimum point of the cost function. These methods have usually a high speed, but may be stuck in local optimum trap instead. In the opposite side, there are methods that begin to search for absolute optimum point for the function. Genetic algorithm and particle swarm optimization are examples for these methods.

The results of experiment show a suggested method on different cost functions that the mentioned algorithm will be successful in finding optimum point for these functions. The different useful problems solved using this algorithm show that the strategy of mentioned optimization can successfully help us to solve useful and engineering problems alongside other mentioned optimization methods such as genetic algorithm and particle swarm algorithm. The result comparison of mentioned algorithm with current optimization methods shows us the relative excellence of mentioned algorithm.

The mentioned optimization algorithm can be used as a simple, quick and proper optimization method for solving most of the optimization methods. Given enough period of time, this method will be converged to a suitable answer. On the other hand, in this optimization method we can simply apply various constraints. In this project, genetic algorithm is used for truss optimization. All the results and figures are created due to high ability of this algorithm in quick convergence and finding optimum answer. Evolutionary methods, have digression of local minimum point ability. In the opposite side, classic optimization methods have more convergence speed. To have both high convergence speed and the ability of not being trapped in optimum local points, there is a common method which is the

![Fig. 17. Optimization results, Matlab and Ansys, output report.](image-url)
combination of evolutionary algorithms with classic optimization methods such as Newton method. Hence, we can combine a mixture of mentioned algorithm with classic optimization methods.

Using this method, we hope to gain better results (in convergence). The presented optimization algorithm can be used as a simple, quick and proper method for solving optimization problems. Given enough period of time, this method will converge to an appropriate answer and we can apply different constraints. The results show that this algorithm has a high ability to converge quickly and find optimum answers.

References


