Further Exploration of Evolutionary Algorithms to Generate Large Irregular Tensegrities

Daniel Casper   John Rieffel (Advisor)

Computer Science Department, Union College

Abstract

Tensegrity form-finding is a complex field relating to the discovery and creation of new tensegrity structures. This process has been approached and practiced from many different angles ranging from mechanical, to algorithmic. When researching the use of evolutionary algorithms to evolve irregular structures, the experimenters were met with success, however the line of interest ended there. I propose recreating the algorithms explored in old studies and attempting to match or improve upon the results presented prior.

Introduction and Background

A tensegrity robot is a type of soft robot that consists of rigid struts and elastic strings. When connected, these struts and cables are put under compression and tension respectively, creating a stable yet resilient structure which is a tensile-integrity (tensegrity). One of the most prevailing problems in tensegrity research is that of designing new unique structures upon which to model their robot.

There are many different avenues through which researchers have explored solving this problem. Methods utilizing machine learning [4], algorithmic approaches, and even applications of static, and algebraic analysis [1, 2] have been investigated. Genetic evolutionary algorithms in particular showed promise for finding valid methods to generate new structures for irregular, large tensegrities however it has gone largely untouched in nearly 15 years.

Question

Are evolutionary algorithms still a viable method of generating unique tensegrity structures and are we able to improve on previous method?

Methods

As I am working off research conducted by Rieffel, Valero-Cuevas, ad Lipson in 2009, the algorithm I have been implementing is mostly the same as the one used then. This is a multi-objective optimization algorithm (Figure 1) which evaluates and optimizes a population based on several different objective parameters.

Input : *pop*, list of tensegrities of size *INIT_POP_SIZE* evaluated by form finding function
Output: *pop*, the *INIT_POP_SIZE*/2 most fit tensegrities of the final generation of the algorithm

```
while num_iterations < max_iterations do
    selection sort tensegrities by objective evaluation scores;
    remove all tensegrities with non-unique objective evaluation scores;
    remove indices n/2 to n = poplength;
    repopulate to INIT_POP_SIZE;
    num_iterations ++;
end
```

Figure 1: Algorithm evolving a population through *max_iterations* iterations of the evolutionary algorithm.

When generating and refilling the population, each tensegrity will be evaluated by a form finding function that will provide information about the fitness of the structure and assign a value based on the given objective functions.

Each tensegrity in a population is built off of a graph that is grown using graph grammars in a map L-System as shown in Figure 2. This allows me to easily generate unique larger, more complex graphs from the same original graph just by changing the grammars. With each generation, new tensegrities are created using mutated versions of existing graph grammars in the population. There are primary mutations as well as crossover mutations. Graphs can also undergo secondary mutations.

Current Status

During this fall term I more thoroughly went through the code base that had been provided to me from the original study by Rieffel, Valero-Cuevas, ad Lipson [3]. In doing so I was able to gain a better grasp on how multi-objective algorithms worked as well as how exactly I would interface the algorithm I was working to recreate and hopefully ultimately improve on, with the form-finding functions I would be using.

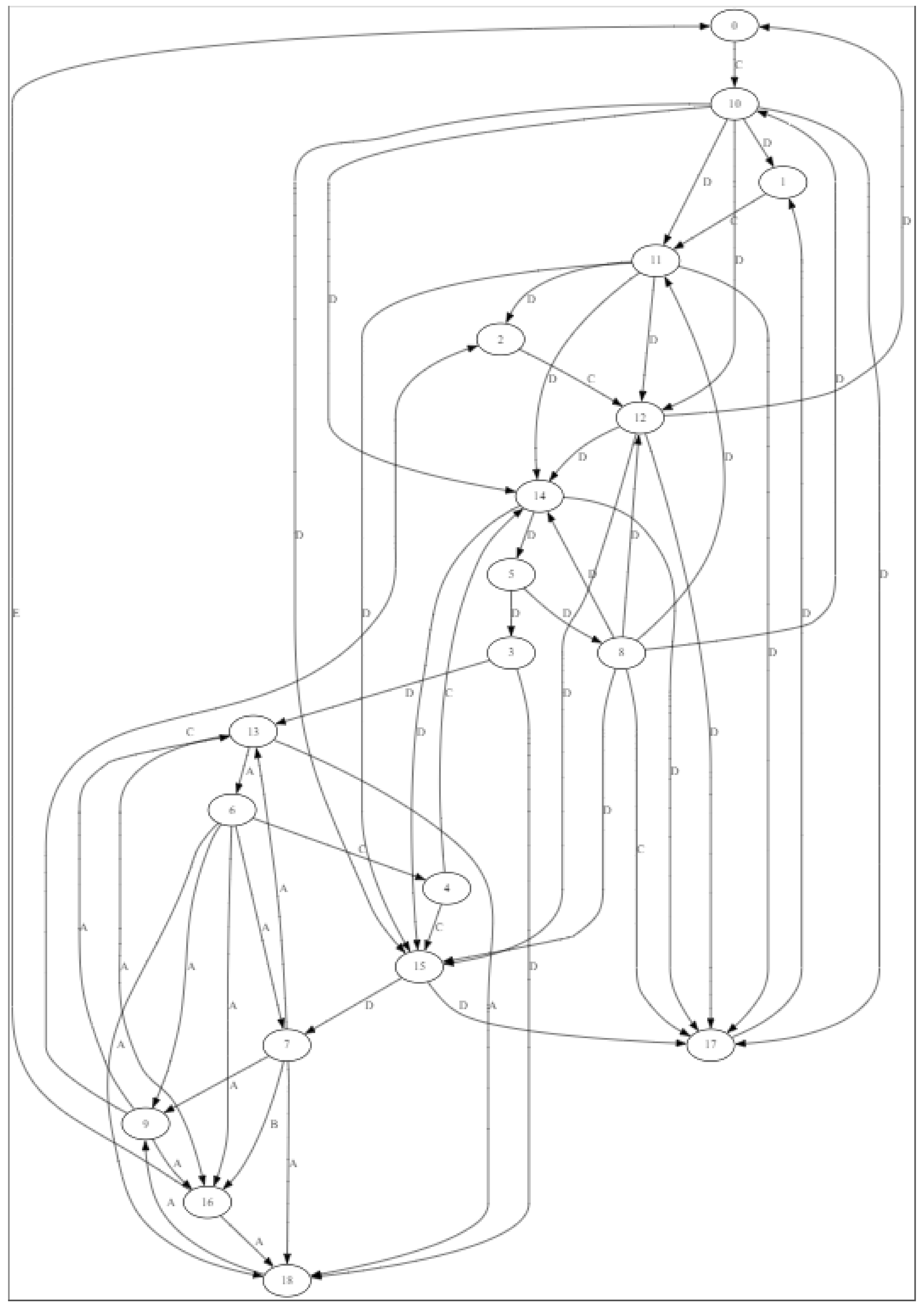

Figure 2: Example of a graph for a basic 3 bar tensegrity after being transformed and grown twice using a map L-System.

As I have now implemented everything but the form-finding function, I will be able to get that sorted out right at the beginning of next term and hopefully begin to start seeing results within the first two weeks. As of now my two options are either Professor Keat's form-finding function or a function provided by David Herrmann who is a colleague of Rieffel's. Both of these options are implemented using MATLAB.

Next Steps

Once the form finding function is integrated, I will be able to begin producing results and running sessions of the algorithm. Comparing the results in these early trials to results from the original study, as well as other studies that have happened since, I will be able to focus in on modifying the objective functions to produce the best results. This can be done by either changing the parameters of the objective or, changing the focus of the objective function entirely.

References

[1] Juan, S. H., and Mirats Tur, J. M.
Tensegrity frameworks: Static analysis review.
Mechanism and Machine Theory 43, 7 (2008), 859–881.

[2] Masic, M., Skelton, R. E., and Gill, P. E.
Algebraic tensegrity form-finding.
International Journal of Solids and Structures 42, 16 (2005), 4833–4858.

[3] Rieffel, J., Valero-Cuevas, F., and Lipson, H.
Automated discovery and optimization of large irregular tensegrity structures.
Computers & Structures 87, 5 (2009), 368–379.

[4] Zalyaev, E., Savin, S., and Vorochaeva, L.
Machine learning approach for tensegrity form finding: Feature extraction problem.
In *2020 4th Scientific School on Dynamics of Complex Networks and their Application in Intellectual Robotics (DCNAIR)* (2020), pp. 265–268.