SimpleOpt: A Simplified Topology Optimization Algorithm

BY MICHAEL MILLER

December 11, 2013

Abstract

Current methods of Topology Optimization use series of differential equations to model stress and strain on millions of infinitesimal elements of a single solid. This process is repeated hundreds of times, following one of many optimization algorithms. Clusters of computers working in parallel can take hours to optimize a small part.

This project sought is to create a simplified version of the Topology Optimization process. A Java RTICA was created which uses only basic concepts of physics and mechanics. The RTICA's toy physics will approximate the results of programs that utilize more sophisticated models of physics.

1 Background

Topology Optimization is a design process in which three-dimensional, load bearing parts are created with minimal material at maximum strength. In theory, the process is relatively simple.

First, the engineer inputs a current part, and the forces that this part will likely undergo. The program then analyzes the stress and strain on each minute element of the part. Areas with very low stress are removed, and areas with high stress are reinforced. The program then iterates, continuing to remove and add material until certain criteria are met. For example, the program might stop once all elements have less than a certain amount of stress and iterating again would push them over this limit.

The results of this process are usually asymmetric, organic-looking parts. In the past, these parts had little use since they are difficult to manufacture. However, in an era of 3D printing, companies are turning more and more readily to Topology Optimization.

Figure 1. An optimized part

2 Program Overview

2.1 Finite Element Analysis

The FEA in SimpleOpt calculates the stress at each element in a predefined, two dimensional grid. The analysis starts with an applied force and follows its path downward until it reaches the ground. Each element that receives force distributes it to the three elements beneath it. Often, an element is positioned in a way that its force can never reach ground. In this case, the element is marked as unsupported and the analysis does not apply force to it.

The location of forces and the location of the ground can be modified by the user. In addition, elements where material is undesirable can be permanently excluded from the analysis.

The actual process of analysis in SimpleOpt is complex and highly recursive. The program analyzes each applied force individually. In order to eliminate redundant calculation, the program works recursively, following the boundary of the propagating force. When the algorithm analyzes an element, it first finds all of that element's children - the elements below it that will receive force.

The algorithm distributes the appropriate amount of force to each child, and then adds it to a list. This process is repeated for each element in the boundary. However, if an element is a child of two different elements in the boundary, it is added to the list only once.

The list will now contain the next boundary, and the process is repeated. By working with boundaries instead of individual elements, the analysis must only consider each part of the boundary once.

In addition, because this system only considers boundaries, it completely ignores row and column numbers. The program can be quickly adapted to propagate forces in any number of directions, not just down. This is important if SimpleOpt is ever to be expanded, as discussed below.

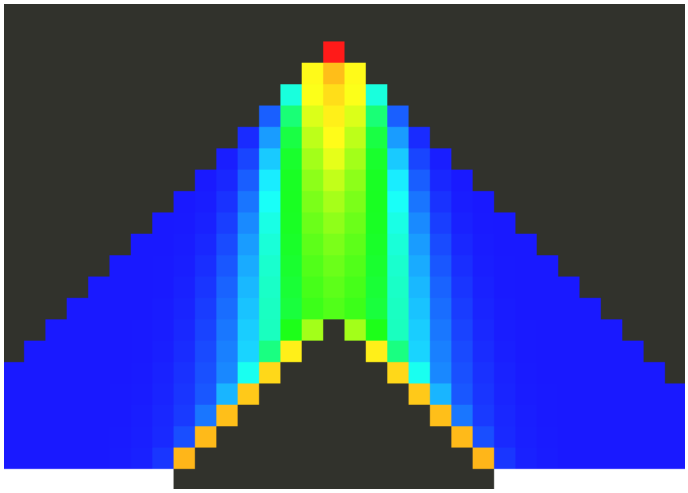

Figure 2. A sample of SimpleOpt's finite element analysis

2.2 Optimization Algorithm

The program uses a Level Set Algorithm [1] (Figure 3) to optimize its material. In this sense

SimpleOpt only regards the stress on the edge of material. The program calculates how many elements are currently in use and compares this to the predefined goal. If there is more material than required, the program finds the element on the boundary with the least amount of stress and removes it. If there is less, the algorithm reinforces the boundary element under the most stress by adding additional material around it.

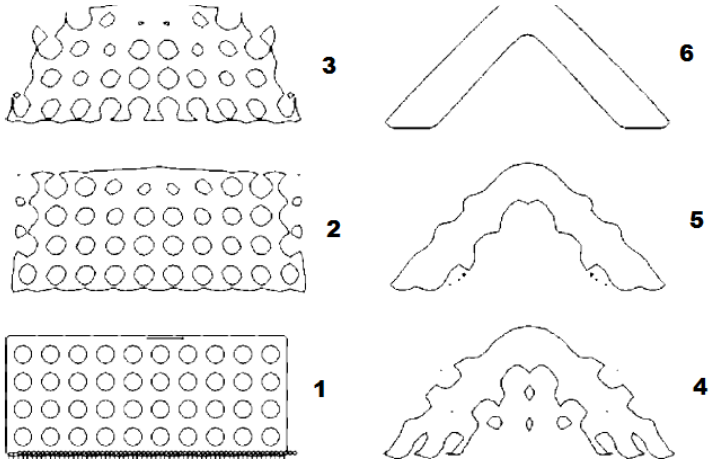

Figure 3. Optimization by a Level Set Algorithm [1]

2.3 Graphical User Interface

The GUI provides the user with a number of options to improve the usability of SimpleOpt. The position and number of forces and ground can be modified and moved, and material can be added to, or excluded from analysis. The target amount of material can be adjusted. In addition to performing at its maximum speed, the analysis can be slowed to allow the optimization process to be visualized.

A significant portion of the code used in the graphical portion of this program comes from Zen Graphics [2]. The options available to the user are based partially off of TopOpt, an iPhone application created by Aage et al, [3].

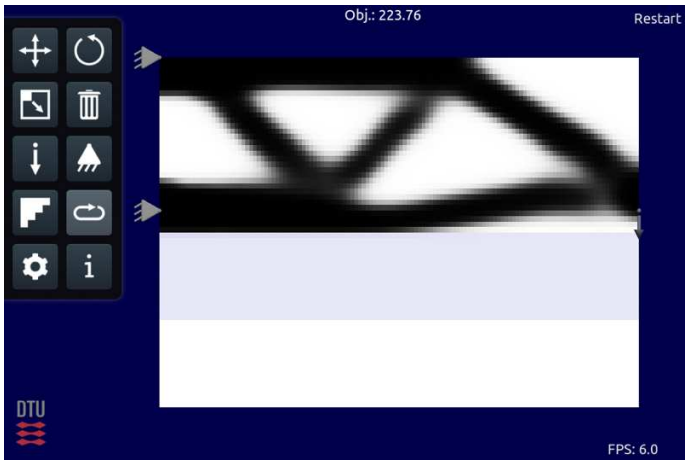

Figure 4. *TopOpt* iPhone screenshot [3]

3 Conclusion

3.1 Reults and Discussion

SimpleOpt can undoubtable perform faster than any optimization software the author has encountered. Because it is not computationally demanding, the RTICA can analyze large numbers of elements, forces, and ground locations with ease. These analyses can be carried out speedily even on weak, personal machines.

However, it must be noted that this program simplifies physics to an extreme degree. By simple inspection it can be seen that this program's optimizations are not at all optimal in the real world (Figure 5). In addition the program lacks a number useful constraints other Topology Optimization programs applied. For example, in SimpleOpt the material is not constrained to be connect, resulting in the creation of parts that are completely separate from each other. For these reasons it is difficult to compare SimpleOpt to other programs.

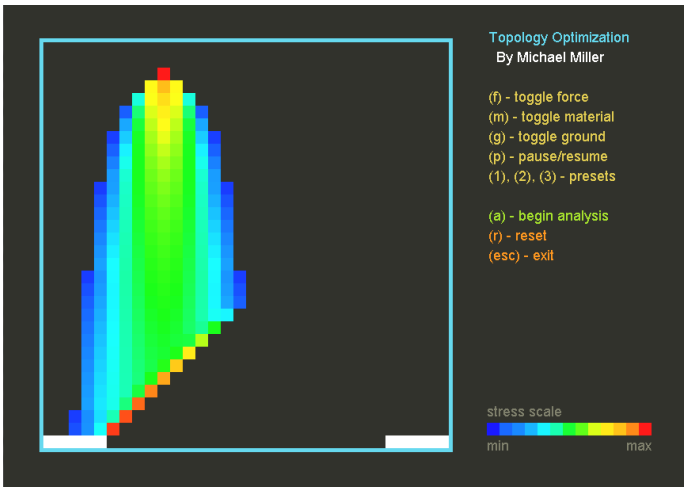

Figure 5. An optimized part that is clearly not optimal under real physics

3.2 Further Study

If additional complexity were added to SimpleOpt, the program could more closely represent physics. However, this additional code could make the program more computationally expensive. It is not known whether adding this complexity would SimpleOpt to run as slowly as other programs.

In order to answer this question, either the algorithms and code behind other optimizers must be further studied and better understood, or SimpleOpt must be expanded and tested.

Should SimpleOpt be improved, it would probably be in the following manner. SimpleOpt's units of stress can be best compared to the y component of forces in physics, so the first step to expand SimpleOpt would be to add an x component of stress. Another addition is the addition of torque, which would further refine SimpleOpt's accuracy towards physics.

Bibliography

[1] Bendsoe, M. P. and Sigmund, O. (2003) *Topology Optimization: Theory, Methods, and Applications*. New York: Springer Publishing.

[2] Angrave, L. (2010). *Zen Graphics* [Java Class]. Publisher: Author. Random House, N.Y.

[3] Aage, Niels, Morten, N. J., Casper, A. S. and Signmund, O. (2013) *Interactive Topology Optimization on Hand-held Devices*. Structural and Multidisciplinary Optimization, 47, 1-6. http://dx.doi.org/10.1007/s00158-012-0827-z

[4] Signmund, O. (2001). *A 99 line topology optimization code written in Matlab* Structural and Multidisciplinary Optimization, 21, 120-127. http://link.springer.com/article/10.1007/s001580050176#page-1