Truss Optimization Using Genetic Algorithms

Andrew Burton
Mathematics and Computer Science, Gonzaga University
Spokane, WA 99258
aburton@gonzaga.edu

Abstract. This paper reports research on the design of structures using genetic algorithms. It presents the design of a program that uses genetic algorithms to optimize a truss structure, along with an example of truss optimization.

1 Introduction

The use of genetic algorithms (GA) for the optimal design of civil engineering structures has been explored in recent years. Ghasemi et al. [1] have demonstrated the suitability of GA to address large trusses with many uncertain variables. This paper illustrates how an algorithm of our design can be used to duplicate this earlier work. Our larger contribution lies in the application of GA to trusses designed under uncertain conditions (Ganzerli et al [2], [3]). Our team[1] is currently writing up the results of research using GA to optimize highly complex trusses.[2]

2 Truss Structures

A truss, found in bridges and roof-supports, is a structure composed of members joined at their end points. See Figure 1. A fundamental engineering task is to minimize the cross-sectional area of the members, while guaranteeing that the truss supports a specified load. This, of course, is an optimization problem. The task is to minimize the overall volume of the truss members while adhering to specified constraints: predicted stresses on the members and the allowable displacements at the joints. Using traditional calculus-based optimization techniques, this is computationally expensive. But traditional techniques have a further drawback. They require that the function used to model the truss be differentiable, and so, continuous. But members are usually off-the-shelf components, available in a range of discrete sizes. The GA allows us to specify truss populations with members drawn from this range.

---

[1] Sara Ganzerli, Department of Civil Engineering, Paul De Palma, and Andrew Burton, Department of Mathematics and Computer Science. Ganzerli and De Palma directed the research presented here.

[2] The research presented in this paper was made possible through a McDonald Work Award, Gonzaga University.

The 10-bar truss, whose solution we illustrate, is shown in Fig. 1. Joints may move only horizontally and vertically. X_i represents the positive displacements, and P_i are the loads. The triangles at Joints A and F indicate that they are externally constrained, and so cannot move. The structural response to the external load consists of three terms: 1) the internal forces of each member, 2) the internal stresses (equal to the internal forces divided by the members' cross-sectional areas) and 3) the displacements. Well-understood matrix relationships allow us to derive structural responses for a given load condition [4].

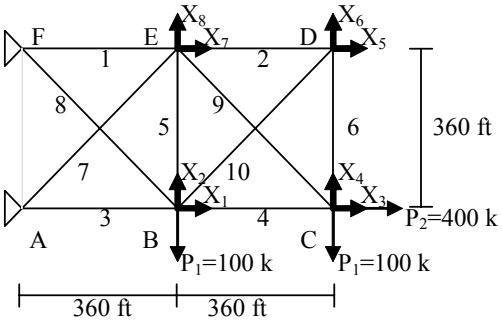

Figure 1. Ten-bar truss

3 Optimal Structural Design Using Genetic Algorithms

GA-Truss is coded in C++, using object-oriented techniques. A generic genetic algorithm class and a parameter class form the base. Derived from these are classes that can use different mating and pairing algorithms. At the next level is a class that calculates a truss's volume and determines whether its constraints have been violated.

The goal is to minimize truss volume—the sum of the cross-sectional areas times the member lengths—with fixed geometry and load conditions. The design process sets constraints on the maximum stresses and displacements, so that the structure's safety and serviceability do not fall below a specified minimum. The optimal design problem for a truss can be stated like this:

$$\text{minimize } f(A, P)$$

$$\text{such that } g_j(A, P) \le 0 \qquad \text{where} \quad j = 1...n$$

where f is the volume expressed as a function of the cross-sectional areas (A) and the external loads (P); $g_j(A,P)$ are the constraints, and n is the number of constraints to be satisfied by the optimal design. If $g_j(A, P)$ exceeds 0 for any constraint, the particular configuration under consideration is unfit.

GA-Truss works with a population of trusses whose members differ in cross-sectional area. Each chromosome in the population is represented as a sequence of member cross-sectional areas, randomly generated. Our cost-function is based on the

structural responses for a given load condition, as defined above. Those trusses that do not meet the constraints are assigned a cost penalty. GA-Truss sorts the population and allows the top half to mate. Parents are paired using a tournament algorithm with a subset size of 20. Three mating algorithms are used, single point crossover, greedy crossover, and static random crossover [5]. One of these is chosen at random at each generation. We begin with a mutation rate of 5%, gradually decreasing to 1%. Convergence is reached when the best truss in the population does not improve over 10 generations.

4 Example: 10-bar Truss

The 10-bar truss of Fig. 1 is a commonly used benchmark in the literature. The constraint is that each member's stress may not exceed 25 kips per square inch (ksi) for both tension and compression, where a kip is 1000 pounds. Member 9 is the exception. Its stress may not exceed 75 ksi. The cross sectional areas are within the range $0.1 \text{ in}^2 < \text{area} < 10 \text{ in}^2$. The truss is aluminum, with a Young's Modulus of 1.0×10^5 psi, where Young's Modulus is parameter describing the stiffness of materials. Ghasemi et al. [1] provides an exact solution to the problem, solved using sequential quadratic programming (SQP), as well as a GA solution. GA-Truss compares favorably to the results published in [1].

Table 1. 10-bar Truss Results

Member	SQP from [1] (in^2)	GA from [1] (in^2)	GA-Truss integers (in^2)	GA-Truss floating (in^2)	GA-Truss rebirth (in^2)
A_1	7.900	7.518	8	7.021	7.437
A_2	0.100	0.458	1	0.980	0.574
A_3	3.900	3.544	4	3.021	3.437
A_4	8.100	8.430	9	8.984	8.576
A_5	0.100	0.100	1	0.100	0.101
A_6	0.100	0.460	1	0.980	0.576
A_7	5.800	6.287	7	7.045	6.469
A_8	5.510	4.992	5	4.274	4.853
A_9	3.680	3.350	4	2.847	3.230
A_{10}	0.140	0.645	1	1.386	0.813
Volume (in^3)	14970	15160	17300	15510	15270

A comparison between the published results and the results obtained with GA-Truss is shown in Table 1.1. The solution of GA-Truss, that is, the overall volume of the truss, is within 2% of the SQP solution. Column 4 presents the situation where design variables were limited to integer values. This reduces convergence time and provides an estimate of an optimal solution. Column 5 is GA-Truss with continuous

design variables. Column 6 presents GA-Truss with continuous design variables using rebirthing. Rebirthing is method of restricting the range of allowable values for the design variables based on their values after a partial run of GA. Since the new population has tighter bounds on the variables it will converge faster. The new ranges are calculated as plus or minus a percentage of the values of each variable after the partial run. A population may be rebirthed several times, with each rebirth giving a tighter bound for the design variables, effectively reducing the search space.

5 Conclusions and Future Work

Here are some generalizations: Two runs might not converge upon the same solution; A larger initial population converges to a better truss; If the mutation rate is set too high, passing on good genes is more difficult; If mutation rate is set too low, the algorithm can get stuck in local minima; A population represented with less precision converges faster; A population represented with more precision converges on a better truss; The precision, or bit depth, can be arranged to allow only integer values for truss member widths. This is useful for a preliminary design. The bit depth may be adjusted for the existing population at run time.

We have also used GA-Truss to solve a 64-bar truss. Further, we have adapted the code to account for uncertainties in the load conditions. Our team is currently experimenting with allowing the uncertainties to vary. The result will be design curves for truss fit that satisfy any level of uncertainty. Of course, there is a tradeoff between the level of uncertainty and the structural cost: the higher the uncertainty, the higher the cost. The design curves should help identify the best possible compromise, providing a useful tool for the engineer.

References

1. Ghasemi, M., et al.. 1999. Optimization of Trusses Using Genetic Algorithms for Discrete and Continuous Variables. *Engineering Computations*. MCB Univ. Press Ltd., Bradford, Engl. Vol. 16 (No. 3): 272-301.
2. Ganzerli, S., De Palma, P., Smith, J., Burkhart, M. 2003. Efficiency of genetic algorithms for optimal structural design considering convex models of uncertainty. *Proceedings of The Ninth International Conference on Applications of Statistics and Probability in Civil Engineering*, San Francisco, July 6-9, 2003.
3. Ganzerli S. and Burkhart M.F. 2002. Genetic algorithms for optimal structural design using convex models of uncertainties. *Fourth International Conference on Computational Stochastic Mechanics* (CSM4), Kerkyra (Corfu), Greece. June 9-12, 2002.
4. Wang, C.K. 1986. *Structural Analysis on Microcomputers*. New York, NY: Macmillan.
5. Haupt, R., Haupt, S., 1997. *Practical Genetic Algorithms*. Wiley-Interscience. Hoboken, NJ.