Leveraging Domain-Expert Knowledge in a Genetic Algorithm for Civil Engineering Design Optimization

[Extended Abstract]

Hoa Vo
Computer Science Dept.
Seattle University
Seattle, WA 98122
voh@seattleu.edu

Justin Terada
Computer Science and Mathematics Depts.
Seattle University
Seattle, WA 98122
teradaj@seattleu.edu

David Joslin
Computer Science Dept.
Seattle University
Seattle, WA 98122
joslind@seattleu.edu

Jeff Dragovich
Civil Engineering Dept.
Seattle University
Seattle, WA 98122
jeffdrag@seattleu.edu

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods, and Search**]: Heuristic methods; J.2 [**Physical Sciences and Engineering**]: Engineering

General Terms

Algorithms

Keywords

Genetic Algorithms, Civil Engineering, Heuristics, Optimization

1. INTRODUCTION

Chromosome evaluation for the civil engineering truss optimization problem is very expensive, thus limiting the number of iterations we can afford to perform. To try to improve convergence we used several mutation operators modeled after strategies used by domain experts. We ran experiments with various combinations of the operators and found that the domain-expert knowledge significantly improves the performance of the GA.

2. IMPLEMENTATION

We implemented a steady-state GA with a population size of 50. Our chromosome consists of a vector of floats representing the sizes of the members in the structure. The fitness is defined by the number of design constraint violations and total weight of the structure. A valid solution has zero design constraint violations. Among valid solutions we want to minimize the total weight.

Each successive generation of the GA is determined by a set of operators that combine and mutate members of the population. We defined four operators:

GECCO'06, July 8–12, 2006, Seattle, Washington, USA.
ACM 1-59593-186-4/06/0007.

Crossover. We defined a standard single-point crossover [3], as well as a crossover that takes the average of the corresponding member sizes.

Mutation. The mutation operator randomly selects a small set of members of the population and increases or decreases the weights. We select about 2% of the members to mutate each time.

Unity Mutation. This operator multiplies each member size by its "unity check" value, a quantity returned by the structural evaluation program. This measures how much stress is on a member. Multiplying a member's cross-sectional area by its unity check value gives an estimate of a better size for the member, increasing its size if it is too small and decreasing it if it is too big.

Directed Mutation. This mutation operator randomly selects a small set of members and increases or decreases the area according to the unity check value. The Unity Mutation modifies every member of the structure, but Directed Mutation attempts to make targeted changes.

3. EXPERIMENTAL RESULTS

With three operators (crossover, unity mutation and directed mutation) there are seven possible subsets. We ran 10 runs of 1000 iterations for each of these seven configurations and averaged the results. On each iteration an operator was selected and applied. Random mutation was implemented as a separate operator and selected with a 5% probability in all configurations.

We began with a structure of 100 members. All of the algorithms found solutions in less than ten iterations. The weights of all seven tests were all very close, but Unity was the worst and GA-DirMut found the lowest weight.

We ran the same tests on a structure with approximately 10000 members. Figure 1 shows the number of constraint violations as a function of the iteration number. Pure GA leveled out with 436 violations. Directed Mutation performed the worst, leveling out at 798 violations. GA-DirMut has the

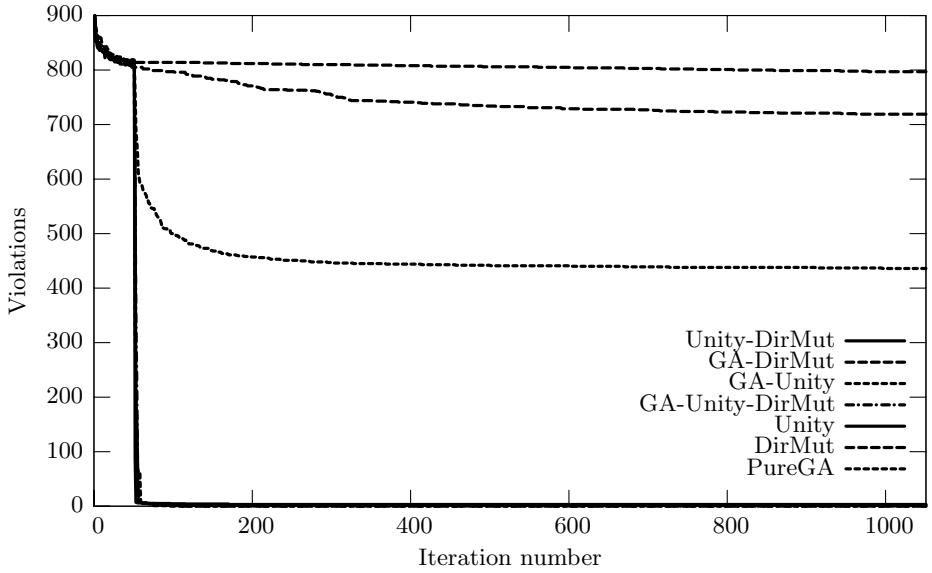

Figure 1: 10000-Member Truss: Total Violations vs Iterations

advantage of recombination with other chromosomes, and leveled out at 718. Unity mutation was the clear winner at early stages in the experiment, initially reducing violations to less than 10 after the initial pool had been filled, although in the end it lagged with four violations. Unity-DirMut was similar, ending with three violations.

Out of all the combinations only two, GA-Unity and GA-Unity-DirMut, were consistently able to find valid solutions, and as the graph shows they tended to find valid solutions after only a relatively small number of iterations. Of these two, GA-Unity was the most effective at reducing the weight of valid solutions.

4. RELATED WORK

Genetic Algorithms have been applied to the sizing of members of trusses, for example in [1]. The largest problem they consider, with 940 members, is an order of magnitude smaller than the problem we use with approximately 10,000 members. Our approach is related to theirs in that they make use of "transformations," although where we use operators that model domain expert methodologies, they use penalty functions to guide the GA. The GA then depends on effective selection of population size, mutation rate, etc., in order to converge on solutions.

Some applications of Genetic Algorithms to civil engineering design have addressed the problem of developing the design model [4, 2]. In this work we only consider the problem of optimizing the sizing of structural members given a fixed geometry. An example of work addressing both aspects of design is found in [5]. Addressing both the geometry and the minimization of member sizes obviously makes the problem more difficult, but the techniques described in that paper are only applied to problems involving significantly fewer members than the larger of the two problems considered.

5. CONCLUSION

Because the fitness evaluation is so expensive for these problems we do not have the option of simply running large numbers of iterations to achieve good quality solutions. We found operators based on domain-expert knowledge to be effective at guiding the search toward a valid solution. Although defining domain-specific operators is not desirable if it can be avoided, when such operators are helpful they can be worth the extra effort. The application of domain-expert knowledge was limited to the design of the operators, and there was no attempt to make "intelligent" decisions about when to apply them. Instead the operators were selected probabilistically, with the GA determining which applications were effective.

6. REFERENCES

[1] M. Ghasemi, E. Hinton, and R. Wood. Optimization of trusses using genetic algorithms for discrete and continuous variables. *Engineering Computations*, 16(3):272–303, 1999.

[2] H. Kawamura and H. Ohmori. Computational morphogenesis of discrete structures via genetic algorithms. *Memoirs of the School of Engineering, Nagoya University*, 53(1), 2001.

[3] M. Mitchell. *An introduction to genetic algorithms.* MIT Press, 1998.

[4] M. Y. Rafiq, J. D. Mathews, and G. N. Bullock. Conceptual building design – evolutionary approach. *Journal of Computing in Civil Engineering*, 17(3):150–158, July 2003.

[5] S. Rajeev and C. S. Krishnamoorthy. Genetic algorithms-based methodologies for design optimization of trusses. *Journal of Structural Engineering*, 123(3):350–358, March 1997.