Evolutionary Algorithm For Structural Optimization

Mark S. Voss
Civil & Environmental Engineering
Marquette University
Milwaukee, WI. 53201-1881
mark.voss@marquette.edu 414-288-6046

Christopher M. Foley
Civil & Environmental Engineering
Marquette University
Milwaukee, WI 53201-1881
c.foley@marquette.edu 414-288-5741

Accepted For Presentation And Publication:

Genetic and Evolutionary Computation Conference (GECCO-99)

July 13 - 17, 1999
Orlando, Florida, USA

A Joint Meeting of the Eighth International Conference on Genetic Algorithms (ICGA-99) and the Fourth Annual Genetic Programming Conference (GP-99)

The 1999 Genetic and Evolutionary Computation Conference (GECCO-99) combines the longest running conference in evolutionary computation (ICGA) and the world's two largest EC conferences (GP and ICGA) to create a unique opportunity to bring together the best in research in the growing field of genetic and evolutionary computation (GEC).

The GECCO conference continues the tradition of the GP and ICGA conferences of bringing together researchers from the entire spectrum of research in evolutionary computation, including genetic algorithms, classifier systems, genetic programming, evolvable hardware, DNA and molecular computing, evolutionary strategies, evolutionary programming, artificial life, adaptive behavior, agents, as well as real-world applications of all of these areas.

Each paper submitted to the GECCO conference was peer-reviewed by one of the independent program committees specializing in various aspects of genetic and evolutionary computation. Each program committee consists of a chair and members who are active researchers and authors of published books and papers in the field of genetic and evolutionary computation. Each program committee will establish its own review criteria and policies and make the final decisions concerning papers submitted to it. These independent "demes" help ensure that the review process respects the diverse traditions and norms of the various facets of genetic and evolutionary computation at the same time it guarantees the acceptance of work of the highest caliber.

Evolutionary Algorithm For Structural Optimization

Mark S. Voss
Civil & Environmental Engineering
Marquette University
Milwaukee, WI. 53201-1881
mark.voss@marquette.edu 414-288-6046

Christopher M. Foley
Civil & Environmental Engineering
Marquette University
Milwaukee, WI 53201-1881
c.foley@marquette.edu 414-288-5741

Abstract

A hybrid rank-based evolutionary algorithm that takes advantage of *a-priori* problem specific information and operates on a high cardinality heuristic genetic representation is presented in this paper. A rank based fitness statement combined with generationally dependant penalty exponents is proposed to condition the seven components of the fitness statement so they participate fairly during the evolutionary process. Translocation crossover and intelligent mutation were utilized to maintain genetic diversity. A graphical method is proposed to monitor the progress of the components of the fitness function, allowing the user to interact with the evolutionary process. Generationally dependant non-linear rank based selection was used to orchestrate a soft landing near the global optimum for an example problem with 20 discrete design variables.

1 INTRODUCTION

The genetic algorithm can be classified as a stochastic procedure and its success depends on the algorithm's ability to effectively search the solution space, while exploiting *good* solutions through genetic reproduction. The generality of the classical genetic algorithm is one of its main assets since it can be applied to a large realm of problems without any *a-priori* problem specific information being required. However, this generality can cause the algorithm to spend time searching regions of the solution space that are known to be unprofitable. A question then arises: *Given two designs for the same structure, what are the structure's building blocks and in what meaningful ways could they be exchanged in the crossover operations typically found in genetic algorithms*: Holland (1975), Goldberg (1989), Holland (1996), Holland (1998).

Traditional multistory buildings tend to have their heaviest members near the base and their lightest members near the top. They also tend to gradually change the weight of their members from floor to floor as one travels up from the base to the roof. In this sense, the building blocks could be seen as the structure's beams, columns and entire floors. This suggests that all corresponding building components (beams, columns and floors) and nearby building components (beams and columns from nearby floors above/below or an entire nearby floor above/below) could participate in meaningful crossover operations. This *a-priori* knowledge of the building blocks contained within the multistory building optimization problem was the motivation for the exploration of a heuristic tree representation for an individual and its design variables.

Representation of design variables in a hierarchical structure suggests reproduction with crossover of genetic hierarchies between mating individuals where crossover occurs at *corresponding* or *nearby* locations. "The easiest way to accomplish this is to introduce an exceptional crossover operator, the translocation operator, which produces crossing-over between randomly chosen non-homologous pairs" Holland (1975). Furthermore, the extension of crossover to higher-order representations (referred to here as macro crossover operations on a heuristic tree representation) was anticipated by Holland (1975) as a means with which to increase the efficiency of the genetic algorithm.

The ability to balance exploration of the solution space while exploiting good solutions is an extremely important attribute for robust GA optimization architectures. The present paper seeks to illustrate a general evolutionary algorithm whereupon the solution space is effectively explored through probabilistic reproduction and fair participation of penalty function components throughout the evolutionary process. Furthermore, exploitation of good solutions is performed through *selection pressure*

applied at flexible stages during the evolution of the optimum solution. Lastly, the paper seeks to present a general, flexible GA architecture that can be applied to a wide range of problems outside the simple cantilever column example provided.

2 PROBLEM DESCRIPTION

The rectangular cantilever column shown in Figure 1 was used to study the effectiveness of the evolutionary algorithm proposed.

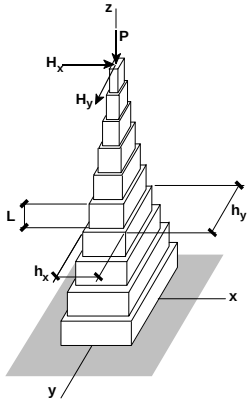

Figure 1: Segmental Cantilever Column

Two design variables (h_x and h_y) are possible for each of the ten segments and the values for these variables are assumed to take on discrete quantities. The derivation and statement of the magnified displacement and stiffness equations can be found in Voss and Foley (1999).

In the current study, the volume of the structure is considered as the characteristic quantity to be optimized. It follows that the optimization problem may be expressed as,

Minimize:

$$V = \sum_{i=1}^{10} \left[L_i - L_{i-1} \right] \cdot h_{x_i} \cdot h_{y_i} \qquad (1)$$

Subject to:

$$\begin{aligned} K_{T_x}^i &\geq K_{T_x}^{goal} & K_{T_y}^i &\geq K_{T_y}^{goal} \\ \delta_{2_x}^{goal} &\geq \delta_{2_x}^i & \delta_{2_y}^{goal} &\geq \delta_{2_y}^i \\ S_x^{goal} &\geq S_x^i & S_y^{goal} &\geq S_y^i \\ h_x^U &\geq h_x \geq h_x^L & h_y^U &\geq h_y \geq h_y^L \end{aligned} \qquad (2)$$

where: V is the volume of the cantilever, and K_i, δ_i, and S_i are the stiffness, deflection, and shape constraints respectively.

In order to employ a genetic algorithm, the above problem statement needs to be reformulated as an unconstrained optimization problem. Constraint functions for a particular individual are given below,

$$\begin{aligned} \Phi_{K_{T_x}}^i &= K_{T_x}^{goal} - K_{T_x}^i & \Phi_{K_{T_y}}^i &= K_{T_y}^{goal} - K_{T_y}^i \\ \Phi_{\delta_{2_x}}^i &= \delta_{2_x}^i - \delta_{2_x}^{goal} & \Phi_{\delta_{2_y}}^i &= \delta_{2_y}^i - \delta_{2_y}^{goal} \\ \Phi_{S_x}^i &= S_x^i - S_x^{goal} & \Phi_{S_y}^i &= S_y^i - S_y^{goal} \end{aligned} \qquad (3)$$

3 ALGORITHM FEATURES

The traditional genetic algorithm works well when one is optimizing multi-variate functions when there is no *a-priori* knowledge of the interrelationships between the solution variables of the objective function. For example, when one knows that the function, $f(a,b,c,d)$ is minimized when, $a \leq b \leq c \leq d$ the traditional genetic algorithm is committed to searching unprofitable regions of the solution universe.

One method to remedy this situation is to modify the penalty functions residing in the fitness statement. Thus, a constraint function of the following form,

$$\begin{aligned} \Phi_{order} = max\left[(a-b),0\right] &+ max\left[(b-c),0\right] \\ &+ max\left[(c-d),0\right] \end{aligned} \qquad (4)$$

could be used to move the population in the direction of promising regions of the search space.

A-priori knowledge of the design variable interrelation also suggests an ordering of the classical genetic algorithm's binary string representation, such that,

$$a \leq b \leq c \leq d \quad \leftarrow goal$$

$$|0001|0101|1000|1001|$$

This provides a basis for hybrid macro crossover operations between individuals whereby two individuals selected for crossover would exchange the value of whole *nearby* variables (translocation crossover). Although translocation crossover operations make intuitive sense in this case, they are not encompassed by the Schemata Theorem; Goldberg (1989).

The present study proposes a problem specific heuristic tree representation; Michalewicz (1992), of design variables as shown in Figure 2. The tree representation

facilitates the recognition of building blocks used in exceptional crossover operations involving homologous and non-homologous pairs and could also be thought of as a genetic program with a static representation; Banzhaf, et al. (1998), Koza (1996). Nodes a, b, and c represent locations for x-dimension, y-dimension and whole level crossover, respectively. The nodes are labeled at the second level, but should be considered as generalized locations for crossover at a given hierarchy level.

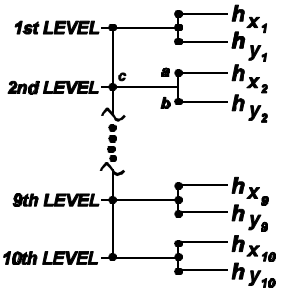

Figure 2: Heuristic Representation

The principal of minimal alphabets (which is supported by the Schemata Theorem), advocates low order genetic representations (low cardinality) for maximum effectiveness; Goldberg (1989). Translocation crossover increases diversity over generations by allowing emigration of variables from one gene location to another. This emigration increases the diversity of potential values that a gene can express with a given genetic representation. If this diversity is maintained, the principal of minimal alphabets may be relaxed; Mitchell (1996).

4 RANK BASED FITNESS

There is a subjective component to optimization in that a designer might be willing to live with a few less than optimal components to get a desired result. Therefore, a highly desirable algorithm would contain options allowing the user to impose the importance of individual components. This, in turn, would give control to a seemingly random optimization procedure. The above considerations motivated the development of a rank based optimization statement. The rank based optimization statement proposed is easily implemented and somewhat semi-automatic. The designer must assign a scalar multiple and exponent to each optimization statement component based on both objective knowledge and subjective preference.

The weight and all constraint components of each individual are ranked with all components less than zero given a rank of zero and the smallest non-negative component(s) given a rank of one with the next largest given a rank of two, etc. Weight and constraint components that have the same value are assigned the same rank. Once the ranks have been assigned they are multiplied by a scaling multiplier and added to a constant. The result is then raised to a Generationally Dependant Penalty Exponent (GDPE) to tune the relative weight of each component in the objective function. For example, given the following numerical values for the displacement constraint violation for an assumed population of seven individuals:

$$\begin{aligned}\Phi_\delta &= \delta^i - \delta^{goal} \\ &= \{13, -52, 25, 2, -1000, 342, 13\}\end{aligned}$$

the ranks of the displacement constraint violations for this population are then defined as,

$$R(\Phi_\delta) = \{2,0,3,1,0,4,2\}$$

The rank of the displacement constraint violation for the sixth individual in the population is denoted,

$$R_6(\Phi_\delta) = 4$$

The component penalty with respect to displacement for the i^{th} individual in the population is defined as,

$$f_i^{\delta} = \left[1 + \xi_\delta R_i(\Phi_\delta)\right]^{n_\delta}$$

where: ξ_δ is the multiplier for the deflection constraint. Similar values are defined for volume, stiffness, and shape. In the present formulation, all values of ξ are 1.0 as indicated in Table 1. For example, the displacement penalty for the sixth individual in the population is,

$$f_6^{\delta} = \left[1 + \xi_\delta \cdot 4\right]^{n_\delta}$$

The rank based fitness for an individual ten segment cantilever column can then be written as follows;

$$F_i = f_i^W + f_i^{K_{Tx}} + f_i^{K_{Ty}} + f_i^{\delta_x} + f_i^{\delta_y} + f_i^{Sx} + f_i^{Sy} \qquad (5)$$

Table 1: User Defined Constants

Constant	No Shape Penalty	Medium Shape Penalty	Large Shape Penalty
α	0.000	0.500	1.000
β	0.000	2.000	2.000
ξ	1.000	1.000	1.000
γ_1, γ_3	0.030	0.030	0.030
γ_2	0.003	0.003	0.003
$\zeta_1, \zeta_2, \zeta_3$	0.190	0.190	0.190
$\lambda_1, \lambda_2, \lambda_3$	3.000	3.000	3.000

5 ALGORITHM COMPONENTS

Generationally Dependent Nonlinear Rank Based Selection; Back (1996), Michalewicz (1992), combined with GDPE(s) was employed to allow dynamic control of algorithm convergence. By ranking the rank based fitness values themselves (each individual given a fitness rank from 1 to the population size) and then using GDNLRBS, it was possible to dynamically control how fast the selection pressure was increased from somewhat egalitarian (low) pressure during early generations, to high (focused on the exploitation of fit individuals) during later generations. It should be noted that GDNLRBS is a global selection tuning mechanism whereas the GDPE(s) are local selection tuning mechanisms. The interrelation of these tuning mechanisms is integral to the efficient operation of the proposed algorithm.

The generationally dependent penalty exponents are defined by,

$$n_i = \alpha + \beta \cdot \left[\frac{G_{curr}}{G_{max}} \right] \qquad (6)$$

where: α and β are user selected constants; G_{curr} is an integer representing the current generation; and G_{max} is an integer defining the maximum number of generations to be carried out in the genetic algorithm. By choosing the values of α and β carefully it is possible to focus the evolutionary search on different penalties at different points during the evolution of the optimum design.

The general form of the selection probability for crossover, carry-over, and mutation is given by:

$$p(r) = \frac{q(1-q)^{r-1}}{\left[1-(1-q)^{m}\right]} \qquad (7)$$

where: r is the rank of the individual; m is the population size; and $q \in (0...1)$. Larger values of q imply stronger selective pressure of the algorithm; Michalewicz (1992). A generationally dependent selection pressure parameter, q, is defined as,

$$q = \gamma + \zeta \left[\min\left(1 , \frac{G_{curr} + 5}{G_{max}} \right) \right]^{\lambda} \qquad (8)$$

where; γ, λ and ζ are user defined constants. All multipliers, constants and exponents used in the proposed algorithm are given in Table 1.

A cumulative probability density function (Figure 3) is constructed using the nonlinear probability function, *p(r)*. An individual can then be chosen by selecting a random real number $r \in (0...1)$ and mapping it onto the cumulative probability density function to determine the rank of the individual to be selected.

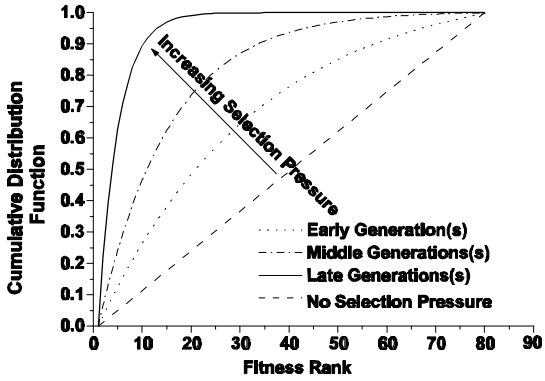

Figure 3: Generationally Dependant Nonlinear Rank Based Selection

Elitism was also employed in the proposed evolutionary algorithm. The top two individuals of every generation are carried into the next generation. Since GDNLRBS is used, the algorithm has a lessened tendency to exploit super-fit individuals than proportional fitness selection. This allows the benefits of elitism to be employed without premature convergence on super-fit individuals during early generations.

Figure 4 illustrates the flow diagram for the proposed hybrid genetic algorithm.

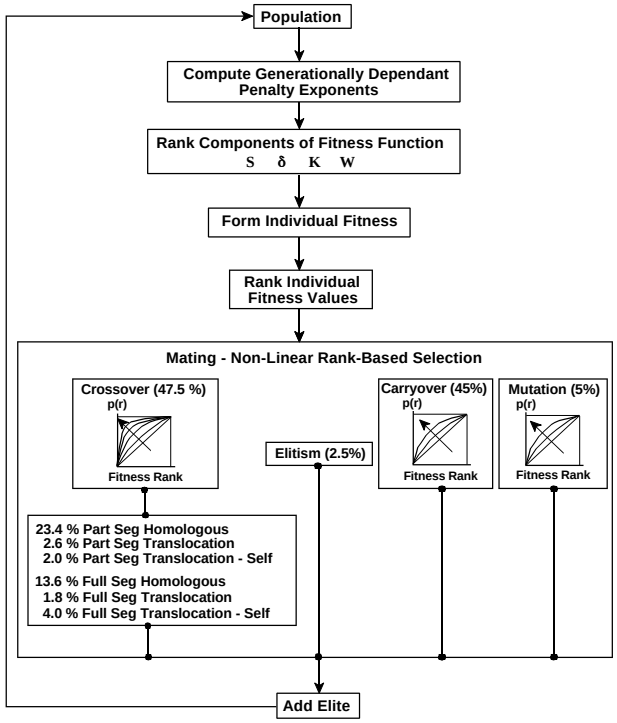

Figure 4: Evolutionary Algorithm Flow Diagram.

Separate GDNLRBS functions were used for selecting individuals for participation in crossover, carryover and intelligent mutation. It was felt that creation of separate GDNLRBS distributions for crossover (ζ_1 , γ_1 , λ_1) , carryover (ζ_2 , γ_2 , λ_2), and mutation (ζ_3 , γ_3 , λ_3) did not overly complicate the algorithm and allows for maximum flexibility with respect to tuning. The three components are used in conjunction with one another to tune the evolutionary algorithm. Percentages for reproduction using GDNLRBS are also given in Figure 4.

In traditional crossover operations it is possible to define the amount of material swapped during reproduction, since there is generally only one type of crossover used (uniform, single/double point, etc.). The present study implemented 6 exceptional crossover operations during reproduction. Due to the complexity of these crossover mechanisms cascading over one another, it is not possible to directly set the amount of genetic material that is swapped during reproduction. It was therefore necessary to run a simulation to determine the percentage of genetic material exchanged; Voss and Foley (1999). The average amount of genetic material swapped between individuals and swapped internally during reproduction was found to be 41.5% and 6% respectively. *This results in a total average reproductive genetic modification from crossover of 47.5 %*.

The crossover operations (operating on generalized nodal locations a, b, and c) are listed and discussed as follows:

1.) ***Homologous Partial Segment Crossover:*** corresponding "a" or "b" locations on two unique individuals are crossed over.
2.) ***Non-homologous (Translocation) Partial Segment Crossover:*** "a" or "b" location crossed over with an "a" or "b" location offset up to four levels away on two unique individuals.
3.) ***Non-homologous (Translocation) Self Partial Segment Crossover:*** "a" or "b" location crossed over with an "a" or "b" location offset up to four levels away on the same individual.
4.) ***Homologous Segment Crossover:*** Corresponding "c" locations on two unique individuals are crossed over.
5.) ***Non-homologous (Translocation) Segment Crossover:*** "c" location crossed over with another "c" location offset up to four levels away on two unique individuals.
6.) ***Non-homologous (Translocation) Self Segment Crossover:*** "c" location crossed over with another "c" location offset up to four levels away on the same individuals.

Figure 4 illustrates the percentages of each crossover operation in the proposed algorithm.

The proposed algorithm does not employ a criteria for termination. Instead, the algorithm uses the GDNLRBS to force convergence after a given number of generations. By simply tuning the convergence parameters, it is possible to orchestrate a soft landing close to the global optimum.

6 RESULTS AND DISCUSSION

The cantilever example problem developed was given a fixed length of 500 inches divided into 10 equal segments of 50 inches. The orientation of the loads and degrees of freedom are shown graphically in Figure 1. Each of the segments could vary between 10 and 80 inches in both the x- and y-directions adhering to discrete increments of 0.35 inches. An axial load of P_z = 80,000 kips combined with horizontal loads of $H_x = H_y$ = 50 kips were applied at the top of the cantilever. The proposed evolutionary algorithm was run for 50 generations with a fixed population size of 80. Table 1 should be referenced for information regarding the evolutionary algorithm parameters used for all runs. The constraints assigned for the problem are given in Table 2 below.

Table 2: Constraint Parameters

Constraint Quantity (Goal)	Magnitude
K_{T_x} , K_{T_y}	0 , 0
δ_{2_x} , δ_{2_y}	2.0 , 20.0
S_x , S_y	0 , 0: refer to equation (4)
h_x^U , h_x^L	10 , 80
h_y^U , h_y^L	10 , 80

Figures 5, 6 and 7 show the component ranking of fitness function components for individuals in the population at various stages in the evolutionary process with varying GDPE magnitudes. The graphs are constructed by plotting the sorted values of the rank based component penalty functions. It should be noted that the independent values do not necessarily correspond to the same individual for all component plots. The plots illustrate the relative contribution a particular component plays in establishing the fitness of individuals throughout the population during the evolution. It is easy to monitor these graphs to determine if any one component is dominating the selection process. The user can easily modify the parameter values associated with a dominating component to reduce its contribution throughout the population. The effects of any

modifications are then observed through the use of these plots.

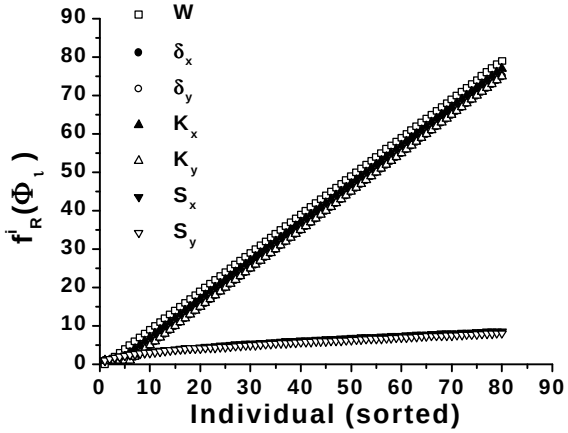

Figure 5: Ranking of Fitness Components - Generation Number 1, Medium Shape GDPE.

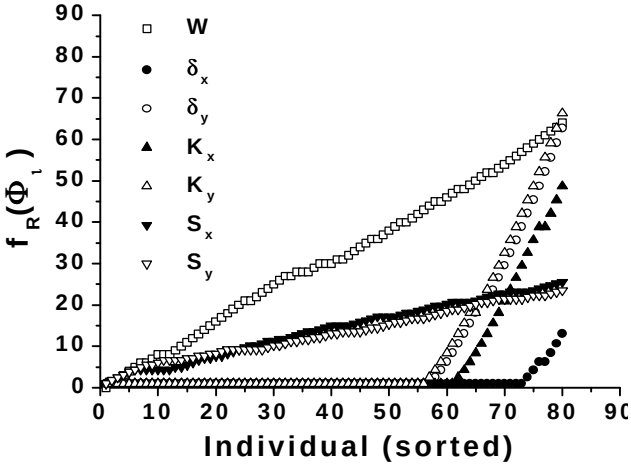

Figure 6: Ranking of Fitness Components - Generation Number 20, Medium Shape GDPE.

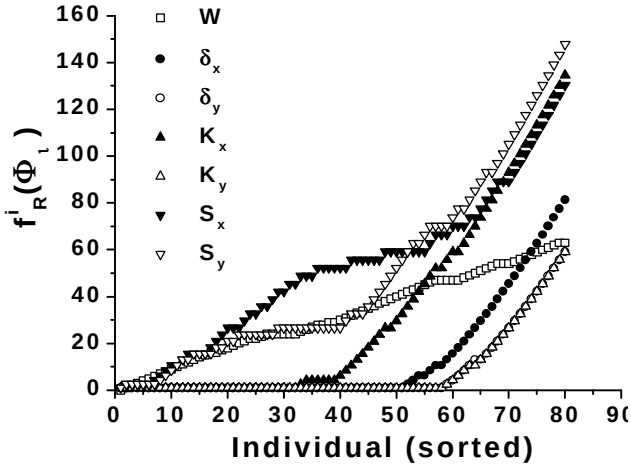

Figure 7: Ranking of Fitness Components - Generation Number 20, Large Shape GDPE.

Figure 5 illustrates that only a few individuals in the initial population are without both deflection and stiffness violations in both directions. Since the value of a component rank for an individual without a constraint violation with respect to that component is zero, the progress of the algorithm is apparent in Figure 6. In this figure, it can be seen that only 24 individuals (of the population - 80 total) have constraint violations with respect to deflection and stiffness penalty components.

Contrasting Figures 6 with Figure 7 it is evident that the large shape penalty exponent is delaying the convergence of the deflection and stiffness components. In Figure 6, the shape components are not so large as to dominate the selection process. When large shape penalties are applied as in Figure 7, the shape components are dominating the selection process. Figure 8 is a plot of the algorithm convergence for the best individual of each generation. The objective of the optimization procedure was to find the minimum volume cantilever meeting all constraints. Therefore, Figure 8 is given as a plot of the generation number versus the volume (in cubic inches) of the best individual of that generation.

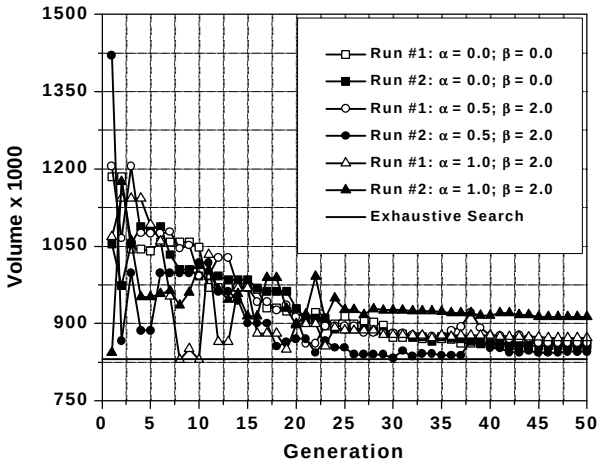

Figure 8: Convergence Trajectories - Best Individual

One needs to be careful when interpreting the convergence plots. The most meaningful information can be inferred from the trajectory of the convergence plot. If very small penalties were applied at the beginning of a run, the volume would be dominant in the selection process. In this case, the convergence plots would approach the global minimum value from below. As the component penalties begin to participate to a larger extent, heavier individuals would become *fitter* than the lighter individuals with high penalty components. This scenario was observed experimentally, but is presented here only as a discussion to aid in interpretation of the convergence plots.

The average volume of the randomly generated initial population also affects the convergence trajectory. A heavy initial population would tend to approach the global minimum from the top whereas a light initial population would tend to approach from the bottom. With this said, the plots presented in Figure 8 are meant to illustrate the relatively good convergence characteristics with respect to the global minimum (found via an exhaustive search procedure). The present algorithm (assuming the step-tapered cantilever configuration) achieved a minimum volume that was approximately 2.8% larger than an exhaustive search procedure (assuming a linearly tapering cantilever configuration). From Figure 8, it can be seen that all runs (with the exception of a large shape penalty run) achieve similar (favorable) results.

Figures 9 and 10 illustrate the importance of translocation crossover and intelligent mutation in the evolutionary process. Convergence trajectories are provided for two separate runs of the algorithm.

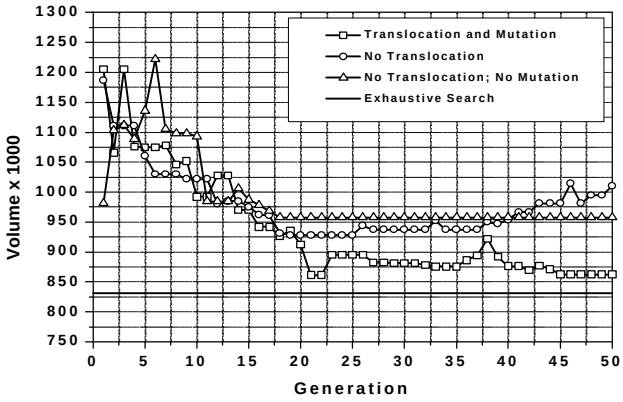

Figure 9: Convergence Trajectories: Run #1 with Moderate Shape GDPE.

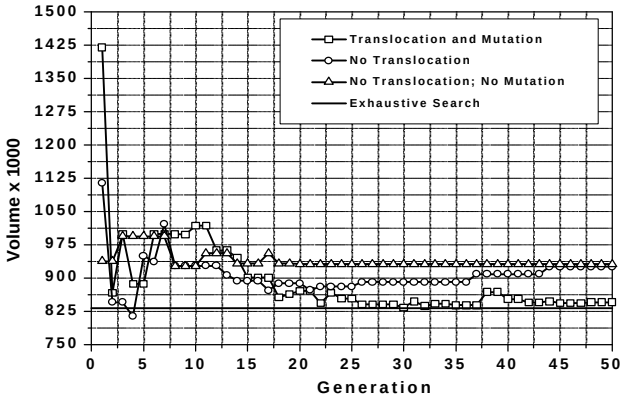

Figure 10: Convergence Trajectories: Run #2 with Moderate Shape GDPE.

It is observed that without translocation crossover, the algorithm reaches a minimum volume at about 25 generations. Since the population has exhausted its diversity around generation 25, the only mechanism that the algorithm has left to combat the increasing penalty exponents is intelligent mutation. The increasing volumes after generation 25 are therefore attributable to intelligent mutation. Any mutation that tends to decrease the shape penalty is accepted. The importance of translocation crossover and intelligent mutation is further illustrated in Figures 9 and 10 where it is observed that without translocation crossover and intelligent mutation, the algorithm runs out of diversity at around generation 20. Figures 9 and 10 together demonstrate the ability of translocation crossover combined with intelligent mutation to maintain diversity during the evolutionary process.

Figure 11 shows the aesthetically pleasing effect of the moderate generationally dependant shape penalty on the final result. The runs with large generationally dependant shape penalties also produced satisfactory designs but they were not as effective with respect to minimizing the weight. This exemplifies the notion that more is not always necessary better in terms of component penalties.

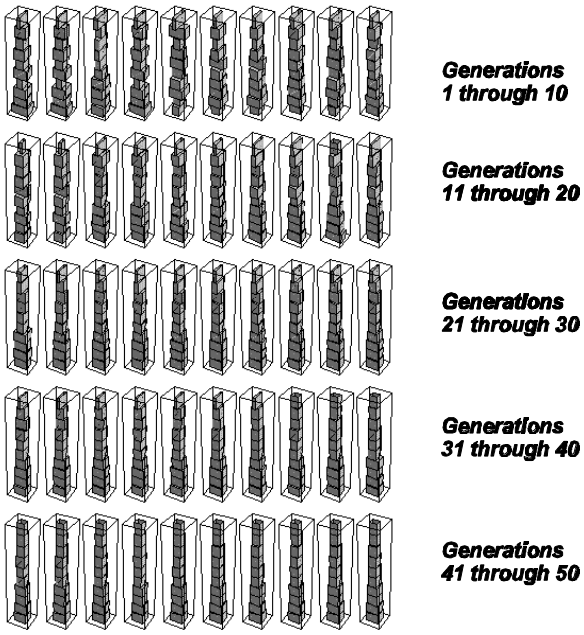

Figure 11: Cantilever Evolution - Medium Shape GDPE.

7 CONCLUDING REMARKS

The proposed evolutionary algorithm meets all of the original design requirements with respect to the theoretical test problem studied here. Given an initial

population of 80 randomly generated 10 segment cantilevers, where each segment's x and y dimension could take on 200 discrete values, the algorithm was easy to setup so that it could consistently come within 3% of the solution found by an exhaustive search in 50 generations. It should be emphasized that only 42 of the 80 individuals need to be evaluated each generation due to carryover and elitism. The results indicate that the proposed evolutionary algorithm will scale well and allows a great deal of flexibility to deal with the complexities of multi-constraint optimization.

A graphical method for interactive algorithm tuning was also developed which allows the user's intuition to be readily incorporated into the selection process. The rate of convergence was graphically demonstrated and easily controlled by modifying components of the algorithm. Therefore, using the proposed algorithm, the rate of convergence could be easily controlled by increasing or decreasing the selective pressure. This gives the algorithm the ability to focus on problem areas without degrading the population diversity which could have detrimental effect with a high cardinality genetic representation such as the heuristic tree used here.

The proposed algorithm has much in common with simulated annealing and was designed such that the final population could be *reheated* thru increased intelligent mutation and/or migration from multiple populations. This, combined with generationally dependent non-linear rank based selection and generationally dependent penalty exponents allows for the implementation of iterative improvement and multiple population evolutionary algorithms which could be used to solve optimization problems where the number of constraints is excessive for a single population to filter.

It should be emphasized that a traditional binary representation is still possible, but may not be necessary depending on the problem and type of translocation crossover and mutation applied.

Acknowledgments
The authors would like to acknowledge the support of the National Science Foundation (USA) - Grant Number CMS 9813216 under the direction of Dr. Priscilla P. Nelson. The views expressed in the paper are those of the authors and not necessarily the sponsor.

References

Back, T. (1996). *Evolutionary Algorithms in Theory and Practice.* New York, New York: Oxford University Press.

Banzhaf, Nordin, Keller, and Francone (1998). *Genetic Programming - An Introduction.* San Francisco: Morgan Kaufmann Publishers, Inc.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning.* New York: Addison-Wesley.

Holland, J. H. (1975). *Adaptation In Natural And Artificial Systems, (An Introductory Analysis With Applications To Biology, Control, and Artificial Intelligence).* Cambridge, London: MIT Press.

Holland, J. H. (1998). *Emergence: From Chaos to Order.* Reading, Mass.: Addison-Wesley.

Holland, J. H. (1996). *Hidden Order: How Adaption Builds Complexity.* New York: Addison-Wesley.

Koza, J. R. (1996). *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge: MIT Press.

Michalewicz, Z. (1992). *Genetic algorithms + data structures = evolution programs.* New York: Springer-Verlag.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms.* Cambridge: MIT Press.

Voss, M. S. and Foley, C. M. (1999), Rank Based Evolutionary Algorithm for Structural Optimization. *Computers & Structures* (submitted for publication).