Parallel Genetic Algorithms in the Optimization of Composite Structures

Erik D. Goodman[1], Ronald C. Averill[2], William F. Punch, III[3], and David J. Eby[2]

Genetic Algorithms Research and Applications Group, Michigan State University

[1]Case Center for Computer-Aided Engineering and Manufacturing, goodman@egr.msu.edu

[2]Department of Materials Science and Mechanics, averill@egr.msu.edu

[3]Department of Computer Science, punch@cps.msu.edu

Keywords: Automated Design, Genetic Algorithms, Composite Material, Laminated Structure, Optimization

Abstract

Genetic Algorithms (GAs) are a powerful technique for search and optimization problems, and are particularly useful in the optimization of composite structures. The search space for an optimal composite structure is generally discontinuous and strongly multimodal, with the possibility for many local sub-optimal solutions or even singular extrema. These facts severely limit gradient-type approaches to optimization, bringing this broad class of problems under scrutiny for application of GAs. Examples described here of the successful use of parallel GAs to design composite structures by the authors include energy-absorbing laminated beams [1], airfoils with tailored bending-twisting coupling [2], and flywheel structures [3]. Optimal design of laminated composite beams was performed using a GA with a specialized finite element model to design material stacking sequences to maximize the mechanical energy absorbed before fracture. An initial GA approach to the optimal design of a specialized, idealized composite airfoil is now being refined for a practical application. The optimum stacking sequence to produce a desired twisting response while minimizing weight, maximizing in-plane stiffness and maintaining acceptable stress levels is determined. The GA has also been used to maximize the Specific Energy Density (SED) of composite flywheels. SED is defined as the amount of rotational energy stored per unit mass. Optimization of SED was achieved by allowing the GA to search for various flywheel shapes and allowing the GA to pick material sequences along the radius of the flywheel.

1. Introduction

Laminated composite construction of panels and other structural elements are used for many applications in the aerospace, automotive, civil, and defense industries. Multi-layer and sandwich construction offer many opportunities for analysts and designers to optimize structures for a particular or even multiple tasks, but this flexibility results in a very large number of often discrete design variables, each with a complex dependence on the others. In addition, the design space may contain many local extrema, and there may be many designs that meet or very nearly meet the design criteria. Thus, there is a need for design techniques that can cost-effectively identify near optimal designs.

Gradient-based optimization techniques have been applied successfully to many shape optimization problems (e.g. [4-6]). However, these methods have several drawbacks. First, they tend to find quickly and get stuck on local extrema [6]. In addition, gradient methods are not suitable for finding singular extrema, or for optimizing truly discrete problems such as the location and geometry of ply dropoffs (terminations) or the material type and orientation of plies in composite structures.

Optimization methods based on genetic algorithms (GAs) have recently been applied to various structural problems [1-3,7-12], and have demonstrated the potential to overcome many of the problems associated with gradient-based methods. However, the need of GAs to evaluate many alternative designs often limits their application to problems in which the design space can be made sufficiently small, even though GAs are most effective when the design space is large. Alternatively, the computational cost per evaluation can be reduced, but that often limits allowable design complexity. A solution to the above dilemma is to perform the GA analysis in a parallel computing environment, where full advantage can be taken of the low communication requirements of GAs, and to use specialized models allowing sufficiently detailed representation without excessive computational requirements.

2. Parallel Genetic Algorithms

Two problems associated with GAs are their computational intensity and their propensity to converge prematurely. An approach that ameliorates both of these problems is parallelization of GAs (PGAs), which also produces a more realistic model of nature than a single large population. PGAs both decrease processing time and better explore the search space. Unlike some specialized sequential GAs which pay a high computational cost for maintaining subpopulations based on similarity comparisons (niching techniques, etc.), PGAs maintain multiple, separate subpopulations which may be allowed to evolve nearly independently. This allows each subpopulation to explore different parts of the search space, each maintaining its own high-fitness individuals and each controlling how mixing occurs with other subpopulations, if at all.

2.1 Coarse-Grain GAs

Coarse-grain GAs (cgGAs) [13] maintain independent subpopulations (often referred to as "islands") which occasionally exchange solutions. The frequency of migration among subpopulations is typically small, and is selected so as to achieve a problem-specific balance between combining good schemata (building blocks) discovered in different subpopulations and allowing the subpopulations to search relatively independently (*i.e.*, promoting diversity). Island parallel GAs have been shown to outperform "serial" GAs dramatically in many contexts [13]. We have categorized cgGAs according to three characteristics: migration method (isolated island, synchronous island, or asynchronous island), connection scheme (static or dynamic), and node homogeneity (homogenous or heterogeneous).

2.2 Injection Island GAs (iiGAs)

We have developed a new PGA architecture called injection island GAs (iiGAs) [13]. iiGAs are a class of asynchronous, static- *or* dynamic-topology, heterogeneous GAs.

The concept of discretizing the topology of a structure for analysis by numerical solution techniques such as the finite element method is by now a familiar one. Often, the initial attempts to analyze a structure involve the use of a coarse discretization (mesh) in which only the critical features of the structure are identified and modeled. In subsequent analyses, progressively finer discretizations may be employed to obtain more accurate local variations of deformation or stress and to determine the effect of these local variations on the overall structural response. In this stepwise approach, additional information about the solution is obtained at each stage of model refinement, and this information is used to guide the development of more refined models, if they are necessary. Often, moderately refined models will provide the majority of needed information, including accurate predictions of the overall structural response and first approximations of local variations.

Where possible, it is both logical and beneficial to take a similar approach in optimal design of structures as well as in other optimization problems. More specifically, the spatial distribution of certain design variables and even the values of design variables themselves can be represented or discretized at various levels of refinement. Optimization using the coarse representations may then provide a gross estimation of a good design that can be improved by performing further optimizations with more refined representations of the structure or its design variables. This is the basis of iiGAs, and their implementation is described below.

iiGA Heterogeneity

GA problems are typically encoded as an *n*-bit string which represents a complete solution to the problem. However, for many problems, the resolution of that bit string can be allowed to vary. That is, we can represent those *n* bits in *n'* bits, *n'*< *n*, by allowing one bit in the *n'*-long representation to represent *r* bits, *r*>1, of the *n*-long bit representation. In such a translation, all *r* bits take the same value as the one bit from the *n'*-long representation and vice-versa. Thus the *n'*-long representation is an abstraction of the *n*-long representation. More formally, let

$$n = p \times q$$

where *p* and *q* are integers, $p, q \geq 1$.

Once *p* and *q* are determined, we can re-encode a *block* of bits *p'* x *q'* as 1 bit if and only if

$$p = l \times p' \ , \quad q = m \times q'$$

where *l* and *m* are integers, $l, m \geq 1$. Such an encoding has the following basic properties,

(i) The smallest block size is 1x1, and the corresponding search space is 2^n.

(ii) The largest block size is *p*x*q*, and the corresponding search space is 2^1=2.

(iii) The search space with a block size *p'*x*q'* is $2^{p/p'} \text{x } 2^{q/q'}$.

An iiGA has multiple subpopulations that encode the same problem using different block sizes. Each generates its own "best" individual separately.

iiGA Migration Rules

An iiGA may have a number of different block sizes being used in its subpopulations. To allow interchange of individuals, we only allow a one-way exchange of information, where the direction is from a low resolution to a high resolution node. Solution exchange from one node type to another requires translation to the appropriate block size, which is done without loss of information from low to high resolution. One bit in an n'-long representation is translated into r bits with the same value in an n-long representation. Thus all nodes inject their best individual into a higher resolution node for "fine-grained" modification. This allows search to occur in multiple encodings, each focusing on different areas of the search space.

More formally, we note that node *x* with block size p_1xq_1 can pass individuals to node *y* with block size p_2xq_2 iff:

$$p_1 = j \times p_2 \ , \quad q_1 = k \times q_2$$

where j, k are integers, $j, k \geq 1$. This establishes a hierarchy of exchange, where node *x* (lower resolution) is the parent of node *y* (higher resolution) and node *y* is the child of node *x*. The direct child is the child with the largest block size; others are "heirs" of the lower-resolution "ancestors." Based on this general migration rule, we have designed the four following static topologies:

- Simple iiGAs
  Each node passes individuals to only one node, the node with the highest resolution (a block size of 1x1).
- Complete iiGAs
  Each node passes individuals to all of its heirs (of higher resolution) in the hierarchy.
- Strict iiGAs
  Each node passes individuals only to its direct children.
- Loose iiGAs
  Each node passes individuals to both its direct children and the node with the highest resolution (block size 1x1).

iiGA Advantages

iiGAs have the following advantages over other PGAs.

(i) Building blocks of lower resolution can be directly found by search at that resolution. After receiving lower resolution solutions from its parent node(s), a node of higher resolution can "fine-tune" these solutions.

(ii) The search space in nodes with lower resolution is proportionally smaller. This typically results in finding "fit" solutions more quickly, which are injected into higher resolution nodes for refinement.

(iii) Nodes connected in the hierarchy (nodes with a parent-child relationship) share portions of the same search space, since the search space of parent is contained in the search space of child. Fast search at low resolution by the parent can potentially help the child find fitter individuals.

(iv) iiGAs embody a divide-and-conquer and partitioning strategy which has been successfully applied to many problems. Homogeneous PGAs cannot guarantee such a division since crossover and mutation may produce individuals that belong to many subspaces, i.e., the divisions cannot be maintained. In iiGAs, the search space is fundamentally divided into hierarchical levels with well defined overlap (the search space of the parent is contained in the search space of the child). A node with block size *r* = *p*' x *q*' only searches for individuals separated by Hamming distance *r*.

(v) In iiGAs, nodes with smaller block size can find the solutions with higher resolution. Although Dynamic Parameter Encoding (DPE) [14] and ARGOT [15] also deal with the resolution problem, using a zoom or inverse zoom operator, they are different from iiGAs. First, they are working at the phenotype level and only for real-valued parameters. iiGAs divide the string into small blocks regardless of the meaning of each bit. Second, it is difficult to establish a well-founded, general trigger criterion for zoom or inverse zoom operators in PDE and ARGOT. Furthermore, the sampling error can fool them into prematurely converging on suboptimal regions. Unlike PDE and ARGOT, iiGAs search different resolution levels in parallel and eliminate the risk of zooming into the wrong target interval, although there remains some risk that search will prematurely converge on a suboptimal region.

2.3 Parallel Computational Environment

The parallel environment in which we first implemented these ideas was based on a modification to GAucsd [16] established by using a modified P4 [17] on a network of Sparc 10 workstations, and has now been more flexibly realized in the GALOPPS [18] toolkit developed by the authors.

3. Composite Applications

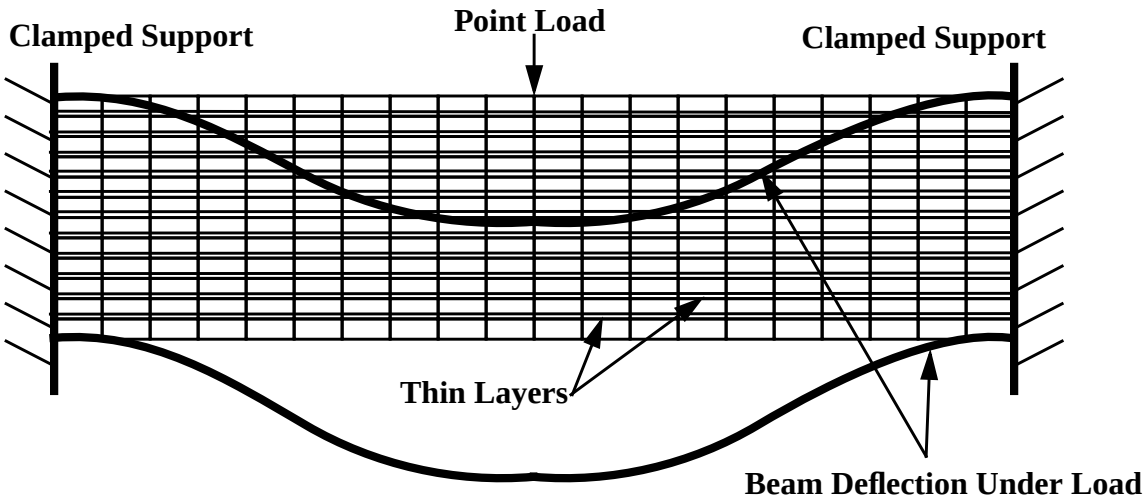


Figure 1. Clamped-clamped laminated beam with center point load.

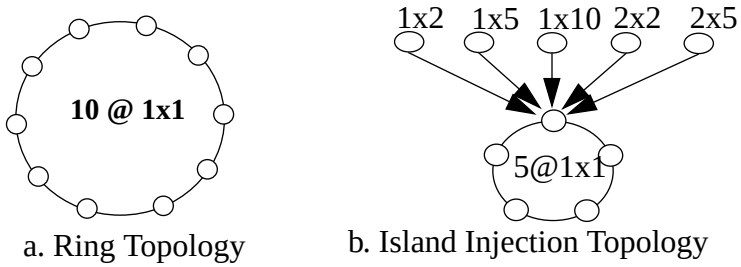


Figure 2. (a) Ring and (b) island injection PGA topologies

3.1 Laminated Beam

The goal of this effort is to develop tools for designing 3D large scale laminated composite panels such as might be found in aerospace, automotive, civil, and marine structures. In order to demonstrate and exercise the capabilities of our models, however, the first application presented here will focus on the design of laminated composite beams, a 2-dimensional problem. In particular, the current application is the design of laminated beams in order to maximize their energy-absorption characteristics, such as might be required in armor plating for tanks or bumpers for automobiles. A simplification is made in that only quasi-static loading is considered.

It is hypothesized that energy absorption can be increased by placing thin compliant layers between the fiber-reinforced composite layers. The purpose of the thin compliant layers is to modify the load path characteristics of the structure for a given applied load to increase the amount of energy the beam can absorb before failure. These compliant layers are a form of “damage” between two layers, and they allow the adjacent layers to “slide” relative to one another under a given load. This sliding mechanism may absorb energy but might also increase the local stresses and reduce the strength of the beam. Thus, the role of the GA is to identify the stacking sequence of the composite layers along with the locations and sizes of the thin compliant layers so as to maximize the amount of energy that can be absorbed before failure. The structural configurations under consideration thus contain complex local material arrangements that significantly affect the local stress/strain state as well as the global deflection response of the structures, both of which are used to evaluate and rank each possible design.

The beam’s energy absorption capacity is determined by finding the maximum load the beam can carry and the associated center deflection before failure occurs, as predicted by a stress-based failure criterion. More specifically, the center deflection and inplane normal stresses in each design element are computed for an applied unit load. These quantities are then scaled linearly to a level at which the inplane normal stress in one or more design elements is equal to the allowable normal stress of the material in that element. The product of the applied load and center deflection at failure, or the work done by the applied load, is then taken as a direct measure of the amount of energy absorbed by the beam before failure.

Results were obtained using both ring topology and island injection topology PGAs. In both cases, ten computational nodes were used, each containing a population of one hundred designs. Thus, the total number of design evaluations per generation in each experiment was the same in each case, allowing a direct evaluation to be made concerning the effect of PGA topology on convergence rate. The node connectivity graphs for the two experi-

ments are shown in Figure 2. In Figure 2b, the topology labeled “island injection” is actually a hybrid topology, with low resolution nodes injecting solutions into a set of high resolution nodes that themselves comprise a ring topology. The differences in convergence characteristics between the two PGA topologies are striking. The results from the ring topology exhibit a classical asymptotic approach to convergence, with indistinguishable differences among the subpopulations.

When using the island injection topology (see Figure 3), converged designs have approximately the same fitness as those obtained using the ring topology. However, the nature of convergence in these two cases is quite different. Referring to Figure 3, it can be seen that, at each stage of the analysis, there is generally a large variation in the fitness of the best individual designs in each node. This is due to the independent search of several nodes at various levels of resolution of the design. More importantly, however, is the rate of convergence, especially during the early stages of the analysis. Consider the number of evaluations required to attain a design with a predicted energy absorption of 5000 Nm (see comments below regarding the predicted energy absorptions). When the ring topology is used, approximately 17,000 evaluations are required to attain this level of fitness, while only approximately 9,000 evaluations are required when the island injection topology is used. The increased rate of convergence at earlier stages of the analysis is even more remarkable, and is easily explained as follows. When a lower resolution of the design is used, the size of the design space is reduced significantly, allowing a faster search for good designs at that resolution. When a node reaches a converged state, it is reinitialized and the search at that resolution is begun anew, as manifested by the “cliffs” on the graph. If the lower resolution discretizations of the design are appropriate --that is, if they are capable of representing good building blocks for the final design -- then nodes searching at these resolutions will produce good designs very rapidly and inject their best designs into the nodes in which the design is represented at a higher resolution, essentially for refinement. In the present case, it is seen in Figure 3 that the node searching at a 2x2 resolution leads the search until approximately 17,000 total evaluations are completed, at which point this node has converged, and the nodes searching at 1x1 resolution are able to find even better designs.

Sample designs are shown in Figure 4. The designs, while slightly different, have very similar levels of predicted energy absorption. This illustrates one of the advantages of genetic algorithms over other design techniques -- several, and possibly many, good designs are generated from which the designer can choose the best one based on other criteria such as manufacturability, cost, etc. Also note the similarity of the designs in Figure 4b-4d to the design in Figure 4a, which was obtained through search at a 2x2 resolution. For reasons of simplicity of design, the beam in Figure 4a may even be the design of choice for a given application. Detailed finite element analyses of selected designs using a commercial finite element code [19] and the current model with a more refined mesh have confirmed the validity of the designs and the model used to obtain them.

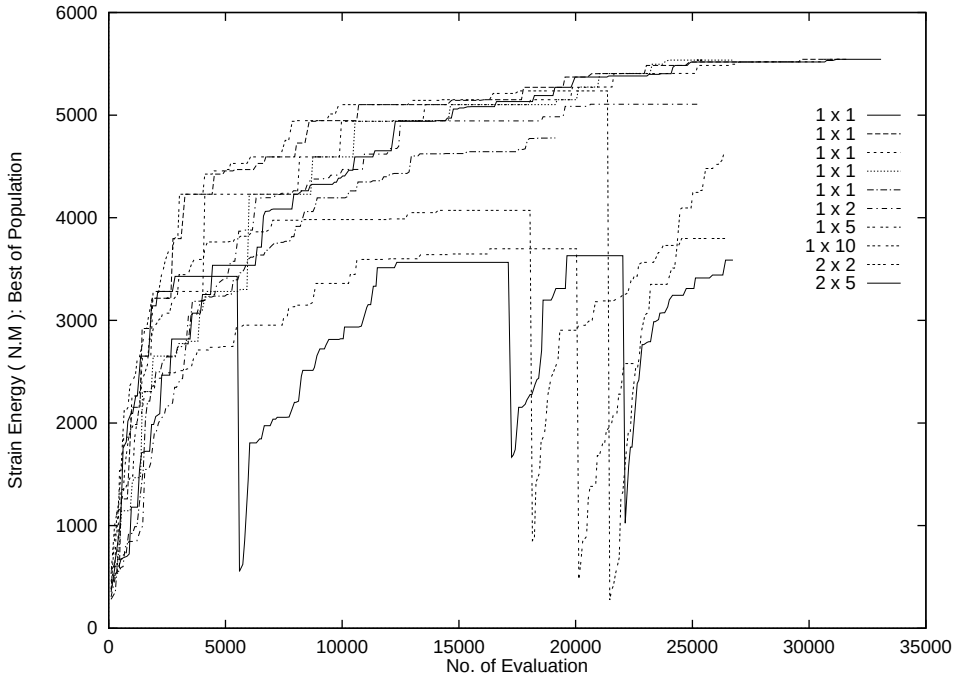


Figure 3. Results of island injection PGA design runs, which yielded superior results to the PGA ring topology. (Sharp cliffs result when converged subpopulations are “restarted”.)

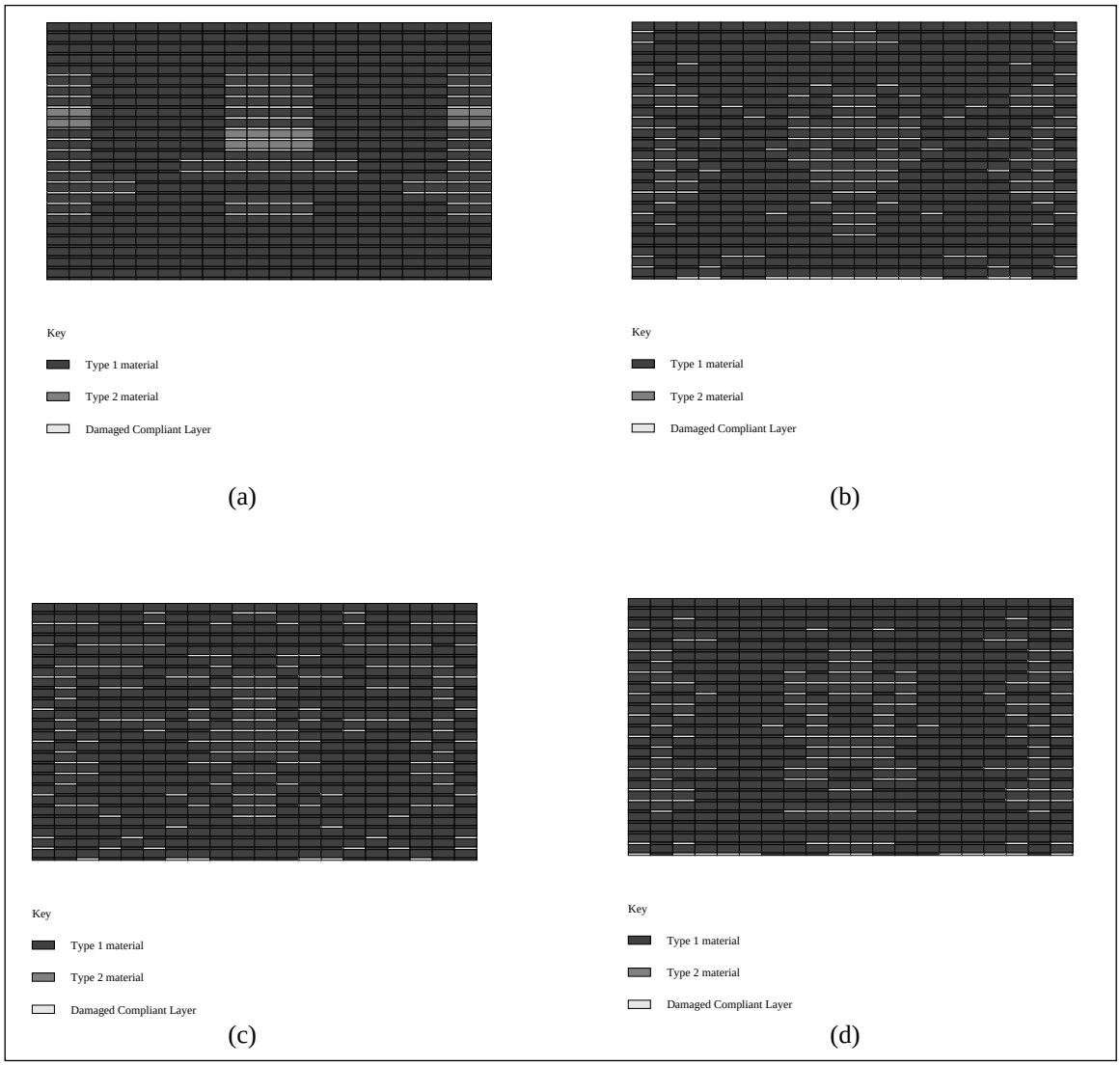


Figure 4. (a) shows best design from the 2x2 subpopulation; (b-d) show best designs from the 1x1 (highest resolution) ring of subpopulations, into which the 2x2 and other designs were injected. 4(b) was the best design produced.

The effectiveness of the island injection PGA architecture for search in moderate-dimensionality (960-bit) design spaces has been demonstrated. Its use is enabled by the availability of a sufficiently accurate and computationally efficient finite element model of a laminated composite beam, since the amount of computation required using a traditional FEA model would still be prohibitively high. The efficiency of the technique shows promise for extension to more realistic design problems involving more complex structures. Such problems can be addressed on a distributed workstation network using the techniques demonstrated here. Our earlier work [13] has showed that, for the problem at hand, using iiGA on a distributed workstation network resulted in approximately linear speedup of the search process, and this fact means that a contemporary workstation network can solve problems one order of magnitude more complex than this one in a timeframe of 1-2 days.

3.2 Composite Airfoil

During flight, aircraft wings are subjected to aerodynamic loading which causes bending and twisting to occur. The twisting load is due to a pressure differential across the airfoil in which pressure is greater at the leading edge than the trailing edge. Many wings are designed with a smaller (typically up to 5 degrees) angle of attack at the tip than at the root ("washout"). Wing twist is built into an aircraft to reshape the spanwise lift distribution to approximate an ellipse and to prevent tip stall [20]. At a given lift coefficient the lift distribution can be optimized by correct choice of initial washout. But when aerodynamic loads cause more pressure to be applied to the leading edge, an upward deflection relative to the trailing edge effectively increases the angle of attack. This work seeks to counteract that effect by engineering a twisting behavior opposite to that caused by the aerodynamic

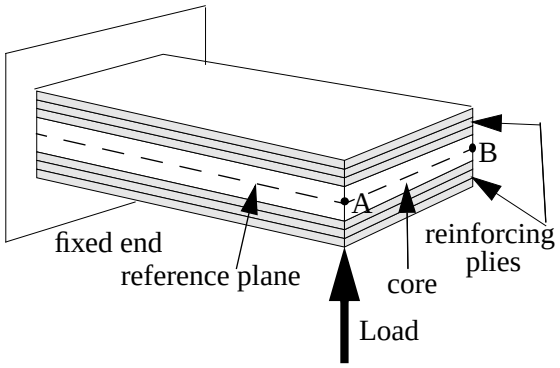


Figure 5. Schematic of the cantilever sandwich panel and loading conditions

loads, so the trailing edge deflects more strongly than the leading edge. This can be achieved using composite sandwich structures in which inherent bending-twisting coupling allows tailoring of the twisting response. For a given loading the objective was to determine the optimum layup (orientation and number of plies in the top and bottom face sheets) which maximizes opposite wing twist while minimizing weight, subject to stiffness and ply clustering constraints. Ultimately, a revised goal will be to achieve a specified amount of opposite wing twist for a given loading condition, rather than to maximize it. The model used is a rectangular cantilever sandwich panel, intended to represent an idealized aircraft wing (see Figure 5). For this demonstration problem, no attempt was made to model exactly actual aircraft wing geometries or loading.

The experiments done compared three different GA topologies -- a single node with a population of 1400, a ring topology with 7 subpopulations having population size of 200 each, and an island injection topology containing 7 subpopulations, each with a population size of 200. In all three cases, a total population size of 1400 is used. Only single population and iiGA results are presented here, due to space limitations. Figure 6 shows single-node results, in which fitness begins to asymptote after about 50 generations. Results from the ring architecture showed each node achieving approximately the same performance after relatively few generations and convergence at around 50 generations. Figure 8b shows the island injection architecture used in this study, and typical results are shown in Figure 7. GALOPPS was used with a two point crossover rate of 0.5, mutation rate of 0.001 per bit, a crowding factor of 3, incest reduction (a form of mate selection) of 3, and tournament selection. The crowding factor and incest reduction were used to help maintain population diversity. Core thickness and angular resolution for the plies in the four top level (lowest resolution) subpopulations is 4 bits, 5 bits for the two middle subpopulations, and 6 bits for the final subpopulation (only one subpopulation at this level was used).

While Figure 2 shows a typical iiGA configuration, a simpler 7-subpopulation tree was used for the preliminary testing (see Figure 8) because of computational resource limitations. In this configuration, the bottom level (highest resolution) contains only one subpopulation into which both moderate level subpopulations inject migrants. Due to the dimensionality of the problem being relatively small, both the iiGA and the ring architecture approaches were able to find a near-optimal solution within approximately 100 generations. Thus additional complexity must be added to the problem to allow conclusive comparison of the different GA architectures. However, certain observations can still be made. Figure 7 shows a typical result of the iiGA, in which migration occurred after each set of 3 generations. The seven curves represent the seven subpopulations. The behavior expected is that the lower-refinement subpopulations, which are working in a smaller search space, will initially make more rapid progress, providing good “building blocks” to the moderate-refinement subpopulations, which should exceed their performance due to a more refined search space. Finally, the most refined subpopulation(s) should receive good building blocks from the moderate-resolution level, and should achieve the best solutions. However, in the small search space represented here, this behavior is compressed into about the first 10 generations. By generation 3, after one generation of “processing” the first migrants from the lowest level, the middle level subpopulations (triangles and asterisks on the graph) have exceeded their performance. Then at about 6 generations, the highest refinement subpopulation (represented by diamonds and dotted line) dominates and continues to outperform the others for the remainder of the run. This run, while outperforming a set of 7 subpopulations at the highest level of refinement using identical parameters and population sizes in a ring topology, reached the same optimum, if allowed to continue to around generation 100. But this example, with only 200 individuals working at the full level of refinement, is eventually beaten by even a single-population GA. However, architectures such as shown in Figure 2(b) do not suffer that problem. The key characteristic of the injection architecture for compute-bound problems is an increased rate of improvement early in the run. For example, note in Figures 6 and 7 that the iiGA produces a design with fitness of 7000 within about 15 generations, while the single node approach

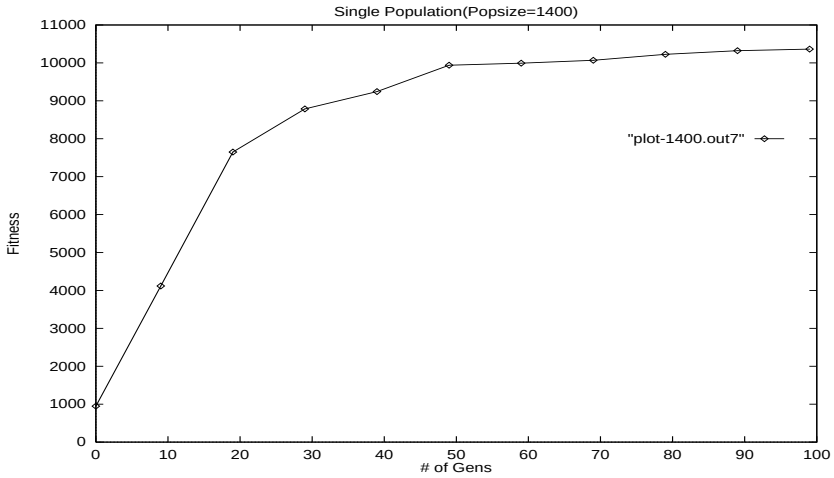


Figure 6. Results from single-node architecture, 1400 individuals.

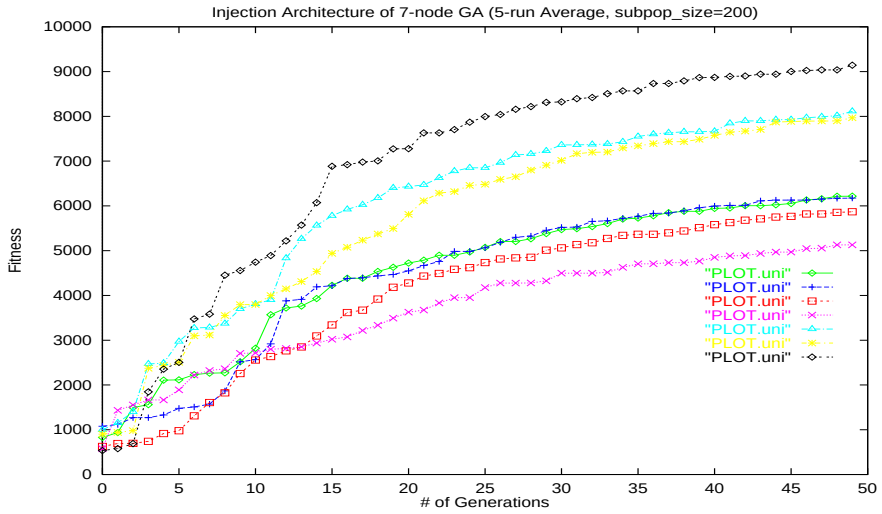
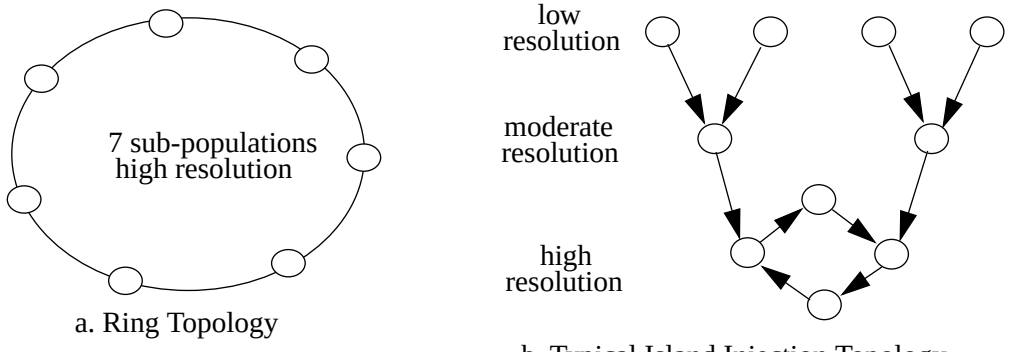


Figure 7. Results from seven-node island injection architecture, each 200 individuals



a. Ring Topology

b. Typical Island Injection Topology

Figure 9. Comparison of levels of refinement and paths for migration among subpopulations

requires at least 20 generations to attain a design with the same level of fitness. Another advantage of the injection architecture is that once the low-resolution subpopulations have converged, they can be either “frozen”, avoiding additional function evaluations, or perhaps partially reinitialized, broadening the search.

3.3 Composite Flywheel

There are numerous applications of flywheels in the area of energy conservation systems. Vehicles can use flywheels during braking for capturing energy lost during deceleration. Another practical application is energy storage in low earth orbit satellites, in which photoelectric cells are exposed to 60 minutes of light, followed by 30

minutes of darkness during which stored energy must be used. The flywheel is well-suited to such applications due to high cyclic lifetimes, longtime reliability and high specific energies. Large flywheels could be used in energy plants.

The fitness function is the Specific Energy Density (SED) of the flywheel, which is the amount of rotational energy stored per unit weight**.** The "fitness" of each flywheel was evaluated with an axisymmetric finite element code that calculates the 3-dimensional static stresses and strains produced in a flywheel at a constant angular velocity. Even for isotropic materials, this model does not make the assumption that the optimal design is one with equal stresses throughout, a common assumption that facilitates analytical solutions. The maximum allowable angular velocity before failure was determined using the maximum stress criterion for isotropic flywheels, and the maximum strain criterion for composite flywheels. The thickness of each ring varies linearly in the radial direction and a diverse set of material choices exists for each ring. A typical annular flywheel is shown in Figure 9.

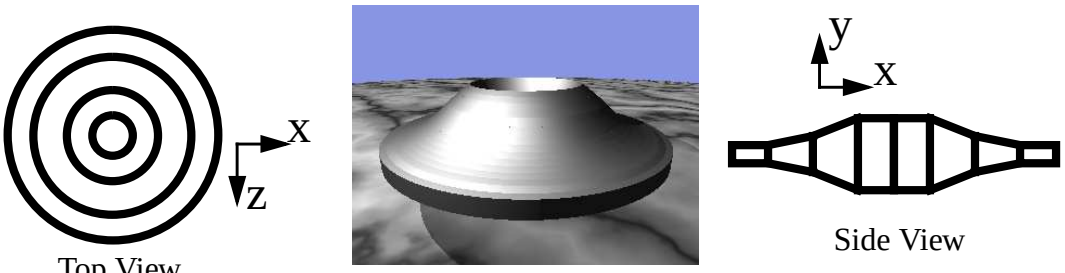


Figure 9. Visual Display of Flywheel

Optimizing isotropic flywheels using the island injection GA seems quite promising. To test the GA, the constant stress shape was first sought. Figure 10 shows that the GA rapidly captures the shape of the isotropic flywheel within 47 generations. Figure 11 compares the GA solution to the analytical constant stress solution. The power of the GA was shown in Figure 12 in the design of a solid isotropic flywheel with stress free edges with an SED that is 55% higher than that of a constant stress flywheel. The commonly used rotor shape of the constant stress flywheel is not optimal when a flywheel has stress free edges. These important findings of the isotropic flywheel can be directly applied to the GA design of composite flywheels. Full results for composite flywheel optimization using island injection GAs will be completed in the near future.

References

[1] Averill, R.C., W. F. Punch, E. D. Goodman, S.-C. Lin, Y. C. Yip, Y. Ding, 1995, "Genetic Algorithm-Based Design of Energy Absorbing Laminated Composite Beams", ASME Design Engin. Tech. Conf., Boston.

[2] B. Malott, R. C. Averill, E. D. Goodman, Y. Ding, W. F. Punch, 1996, "Use of Genetic Algorithms for Optimal design of Laminated Compostite Sandwich Panels with Bending-Twisting Coupling", AIAA/ASME/ASCE/AHS/ASC 37th Structures, Structural Dynamics and MAterials Conference, Salt Lake City, Utah.

[3] D. Eby, R. C. Averill, W. Punch, O. Mathews, E. Goodman, 1997, "An Island Injection GA for Flywheel Design Optimization", Proc. EUFIT '97, Aachen, Germany (forthcoming).

[4] C.A. Soto and A.R. Diaz, 1993, "Optimum layout and shape of plate structures using homogenization," in *Topology Design of Structures*, M.P. Bendsoe and C.A. Mota Soares, eds., pp. 407-420.

[5] K. Suzuki and N. Kikuchi, 1990, "Shape and topology optimization by a homogenization method," in *Sensitivity Analysis and Optimization with Numerical Methods*, AMD-Vol. 115, ASME, pp. 15-30.

[6] K. Suzuki and N. Kikuchi, 1991, "A homogenization method for shape and topology optimization," *Comp. Meth. Appl. Mech. Eng*., **93**, 291-318.

[7] E. Sandgren, E. Jensen, and J.W. Welton, 1990, "Topological design of structural components using genetic optimization methods," in *Sensitivity Analysis and Optimization with Numerical Methods*, S. Saigal and S. Mukherjee, eds., AMD-Vol. 115, ASME, pp. 31-43.

[8] C.D. Chapman, K. Saitou, and M.J. Jakiela, 1993, "Genetic algorithms as an approach to configuration and topology design," in Proc. *1993 ASME Design Automation Conference*, Albuquerque, New Mex., Sept.

[9] S. Nagendra, R.T. Haftka, and Z. Gurdal, 1992, "Stacking sequence optimization of simply supported laminates with stability and strain constraints," *AIAA Journal*, **30**, 2132-2137.

[10] R. LeRiche and R.T. Haftka, 1993, "Optimization of laminate stacking sequence for buckling load maximization by genetic algorithm," *AIAA Journal*, **31**, 951-956.

[11] S. Nagendra, R.T. Haftka, and Z. Gurdal, 1993, "Design of blade stiffened composite panels by a genetic algorithm approach," in *Proceedings of the 34th AIAA/ASME/AHS SDM Conference*, La Jolla, CA, April 19-22, pp. 2418-2436.

[12] M. Leung and G.E. Nevill, Jr., 1994, "Genetic algorithms for preliminary 2-D structural design," in *Proceedings of the 35th AIAA/ASME/AHS SDM Conference*, Hilton Head, SC, April 18-20.

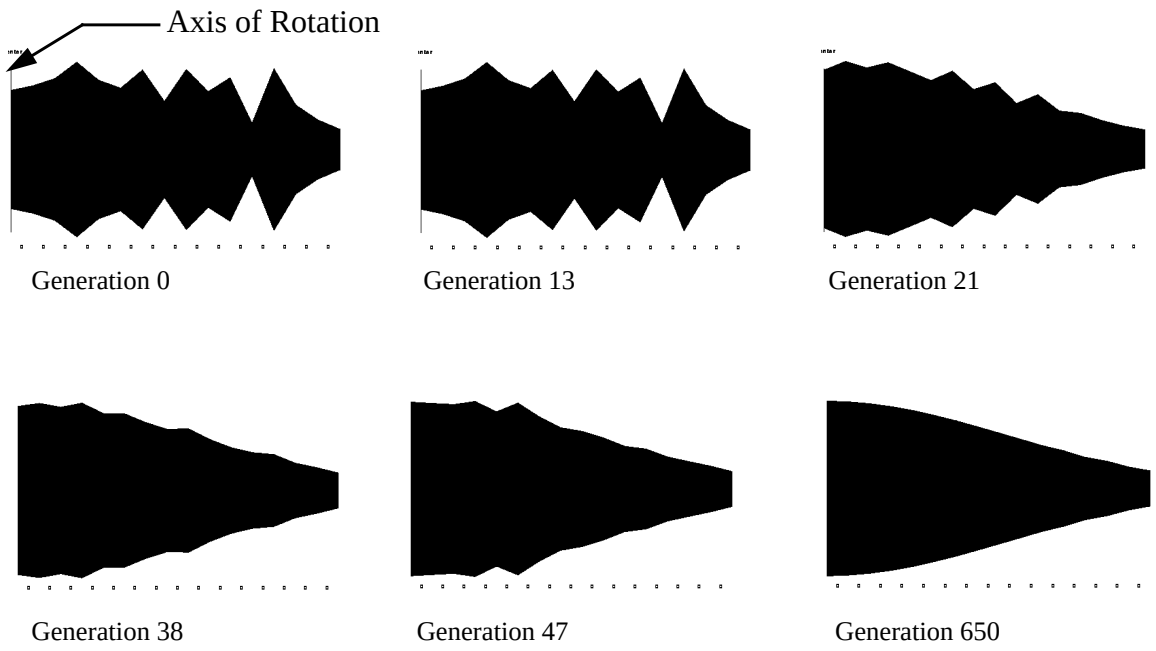


Figure 10. The Evolution of the Constant Stress Flywheel

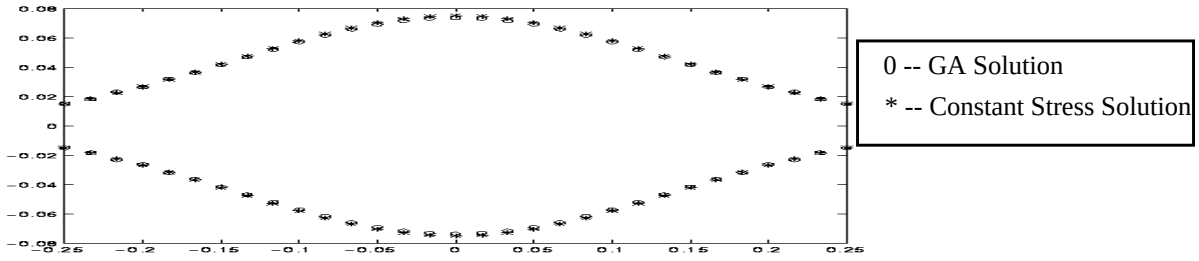


Figure 11. GA Vs. Constant Stress Solution

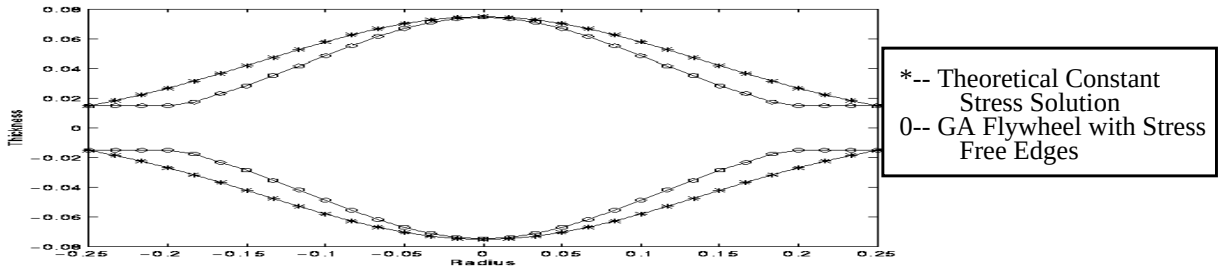


Figure 12. Constant Stress Vs. GA Designed Flywheel with Stress Free Edges

[13] S.-C. Lin, W.F. Punch, and E.D. Goodman, 1994, "Coarse-grain parallel genetic algorithms: categorization and analysis," IEEE Symposium on Parallel and Distributed Processing, pp.27-36.

[14] N. Schraudolph and R. Belew, 1992, "Dynamic Parameter Encoding for Genetic Algorithms," *Machine Learning*, June, pp. 9-21.

[15] C. G. Shaefer, 1987,"The ARGOT Strategy: Adaptive Representation Genetic Optimizer Technique," *Proceedings of the Second International Conference on Genetic Algorithms*, July, pp. 50-55.

[16] N. Schraudolph and J. Grefenstette, 1992, *A User's Guide to GAucsd 1.4*, July.

[17] R. Butler and E. Lusk, 1992, *User's Guide to the P4 Programming System*.

[18] E. Goodman, 1994, GALOPPS, *The Genetic ALgorithm Optimized for Portability and Parallelism System*, Tech. Rept. #94-5, MSU GARAGe, Michigan State University, 100pp.

[19] MARC User's Guide: User Information, 1994, MARC Analysis Research Corporation, Palo Alto, CA.

[20] Yang, P.C., Norris, C.H., and Stavsky, Y., 1966, Elastic Wave Propagation in Heterogeneous Plates, *International Journal of Solids*, Vol. 2, pp. 665-684.