ence on Genetic Algorithms, June 1993, pp. 557-564.

[30] N. Schraudolph and J. Grefenstette, “A User’s Guide to GAucsd 1.4”,Technical Report CS92-249 from Technical Reports, CSE Department, UC San Diego, La Jolla, CA 92093-0114, 1992.

Design", *International Conference on Genetic Algorithms*, June 1993, pp. 408-415.

[11] M. Di Sciuva, "An Improved Shear-Deformation Theory for Moderately Thick Multilayered Anisotropic Shells and Plates," Journal of Applied Mechanics, Vol. 54, pp. 589-596, 1987.

[13] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison - Wesley Publishing Company, Inc., 1989.

[15] P. Hajela, "Genetic Search - An Approach to the Nonconvex Optimization Problem", *AIAA Journal*, Vol. 28, No. 7, pp 1205-1210, 1990.

[16] P. Hajela and C.Y. Lin, "Genetic Search Strategies in Multicriteria Optimal Design", *Proceedings of the AIAA/ASME/ASCE/AHS/ASC 32nd Structures, Structural Dynamics and Material Conference*, Baltimore MD, April, Part 2, pp 354-363.

[17] J. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

[18] C.L. Karr and D.E. Goldberg, "Genetic Algorithm Based Design of an Air-Injected Hydrocyclone", *Control 90, Mineral and Metallurgical Processing*, 1990, pp. 265-272.

[20] R. M. Kling and P. Banerjee, "Empirical and Theoretical Studies of the Simulated Evolution Method Applied to Standard Cell Placement," *IEEE Transactions on CAD*, Vol. 10, No. 10, October 1991.

[21] N. Kogiso, LT. Watson, Z. Gürdal, R.T. Haftka and S Nagendra, "Minimum Thickness Design of Composite Laminates Subject to Buckling and Strength Constraints by Genetic Algorithms", in Proceedings of the *35th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Hilton, S.C., April, 18-20, pp 2257-2275, 1994.

[22] R. Le Riche and R.T. Haftka, "Optimization of Laminate Stacking Sequence for Buckling Load Maximization by Genetic ALgorithm", AIAA Journal, Vol 31, No 5, pp 951-956, 1993.

[23] M. Leung and G.E. Nevill Jr, "Genetic Algorithms for Preliminary 2-D Structural Design", *AIAA 94*, pp 2287-2291.

[24] S-C Lin, W.F. Punch III and E.D. Goodman, "Coarse-Grain Parallel Genetic Algorithms: Categorization and Analysis", accepted to Parallel Distributed Processing 1994 (Taiwan).

[26] W.J. Mitchell, J.P. Steadman and R.S. Liggett, "Synthesis and optimization of small rectangular floor plans, Environment and Planning, 3, 1976, pp. 37-70.

[27] A. Newell and H.A. Simon, *Human Problem Solving*, Prentice-Hall, 1972.

[28] W. Punch, E. Goodman, M. Pei, C.-S. Lai, P. Hovland and R. Enbody, "Intelligent Clustering of High - Dimensionality Data Using Genetic Algorithms," *International Confer-*

tation of the same problem. Such a combination appears to aid both in computational speed up, increased quality of results, and the avoidance of local minima (maintenance of adequate diversity).

7.4 Future Work

There are a number of areas we are currently exploring. The first is an extension of the simulation model to a more accurate 3D model for dealing with plates and shells. The second is an exploration of methods to extend the representation to include structural features like stiffeners and more complicated geometries to increase the complexity of the designs and to more closely model real design practice.

Acknowledgement

The partial support of this work by the State of Michigan Research Excellence Fund is gratefully acknowledged. We also acknowledge the comments of two anonymous IEEE Expert reviewers.

Bibliography

[2] R.C. Averill, “Static and Dynamic Response of Moderately Thick Laminated Beams with Damage,” *Engineering Composites*, Vol. 4, pp. 381-395, 1994.

[3] R.C. Averill and Yip, Y.C., “Development of Simple, Robust Finite Elements Based on Refined Theories for Thick Laminated Beams,” submitted for publication, 1994.

[4] B.G. Buchanan and E.A. Feigenbaum, “DENDRAL and Meta-DENDRAL: their applications dimension, *Artificial Intelligence*, 11(1), 1978, pp. 5-24.

[5] Butler and L. Ewing, “User’s Guide to the p4 Programming System’” Technical Report ANL-92/17, Argonne National Laboratory, Argonne, IL, 1992.

[7] J. P. Cohoon, S. U. Hegde, W. N. Martin and D. S. Richards, “Distributed Genetic Algorithms for Floorplan Design Problem,” *IEEE Transactions on CAD*, Vol. 10, No. 4, April 1991, pp. 483-492.

[9] R.D. Coyne, M.A. Rosenman, A.D. Radford, M. Balachandran and J.S. Gero, *Knowledge-Based Design Systems*, Addison-Wesley Publishing Company, Inc., 1990.

[10] L. Davis, D. Orvosh, A. Cox and Y, Qiu, “A Genetic Algorithm for Survivable Network

view a GA as a good “flaw finder” or “model breaker.” Since a GA blindly (i.e., without prejudice) produces possible designs and feeds them to an evaluation function, it can often uncover flaws in the evaluation function that had never occurred to the designer of the function. Furthermore, because it literally feeds millions of solutions to such a function, it tests many aspects of such a function.

It turns out that the results we obtained, though meaningful from our GA testing point of view, were not as accurate from a mechanical design point of view as would be liked. The GA uncovered some shortcomings in the laminate analysis code *for the current application*. We are addressing those concerns (by means of improved structural models) as our work continues.

7.2 Representation Issues

We have experimented with a number of different representations of 2D and 3D beams as reported here, as well as quite a number of “bad” representations that, in the end, showed us the representations that were most interesting. For the 2D results, it appeared that a 480-bit representation was most appropriate. It was detailed enough to insure good search without wasting computation time. The 3D representation was more interesting, reducing the complexity of the representation in a way that increased manufacturability, while at the same time including 3D information on the cross-section of the beam. We have experimented with a number of encodings of this information onto the chromosome, and the interleaved format as described in Section 4.4 was clearly superior. Again, this is due to the contiguity of information on the GA solution string. All information concerning each layer was contiguous, giving the GA the opportunity not only to find good solutions, but to keep them around while exploring configurations of other layers in the design.

7.3 Parallelization of GAs

The parallelization is the work that has had the most impact on our research. Without it, we would not have been able to explore the rich set of representations we did in a real-world domain. Remember that our original 960-bit representation, summarized in Table 1, ran for more than 9 *days* without achieving the quality of results we did using *any* of the parallel architectures and in considerably less time. Our most promising results are those obtained with the island injection architecture. The iiGAs combine island parallel search with search at multiple levels of represen-

Shown in the figure are the normal 2D beam design material configurations, as well as another view showing the widths assigned to each layer. The dashed lines on the end view indicate what the width of the beam would be if all the layers had the same width, i.e., the average width. The experimental details of the run are shown in Table 4. An island parallel architecture was used in solving the problem, with 10 nodes arranged in a ring exchanging the best solution every 20 generations.

The results indicated that the design was evolving with an “I-beam” cross-sectional configuration, which is what would be expected as the optimal cross-section for these conditions. However, it is interesting to observe that the design with the “better” I-beam cross-section found in the earlier result (left side of Figure 7) is less fit than the later design, which appears to have deviated further from a true I-beam configuration. However, the procedure had not yet converged, so this situation might not have persisted had the run continued. Furthermore, additional study is required to understand the complex relationship between the placement of compliant layers and the cross-sectional shape as regards maximizing energy absorption.

Table 4: 3D Beam Results

Experiment	Number of Nodes	Evaluations per node	Machine Type	Ring Exchange Rate	Max Strain Energy
Figure 7, left side	10	40,000	Sparc 10	every 20 generations	7.355e+03
Figure 7, right side	10	120,000	Sparc 10	every 20 generations	1.034e+04

7. Discussion

7.1 Accuracy of the Simulation Code

An interesting phenomenon that commonly occurs when using genetic algorithms, at least in our experience, is the severe testing, and sometimes “breaking”, of complicated evaluation functions. We originally began work with a laminated beam simulation code that had been tested and appeared robust for a wide range of laminates [2,3]. However, a genetic algorithm design process is a much more rigorous test of such a code than is typically obtained elsewhere. In fact, one can

There remained only the choice of how to configure the GA string to contain the 360 bits required for each design solution. The most logical choice from a genetic algorithm point of view was an *interleaved encoding* which alternates an 11 bit code (1 for material, 10 for the compliant layer element assignments to each element in the half layer) for the material layout of a layer with a 4 bit code indicating the width of that layer, repeated for all 24 layers. In this way, all the information necessary for encoding each layer is contiguous on the GA solution string. Such contiguity is important in a GA encoding. If interdependent solution elements are not located “close-by” on the string, then it makes genetic search difficult as cross-over can easily disrupt good solutions. The chances are then greatly reduced that good solution pieces can co-evolve and remain part of the “best” solution.

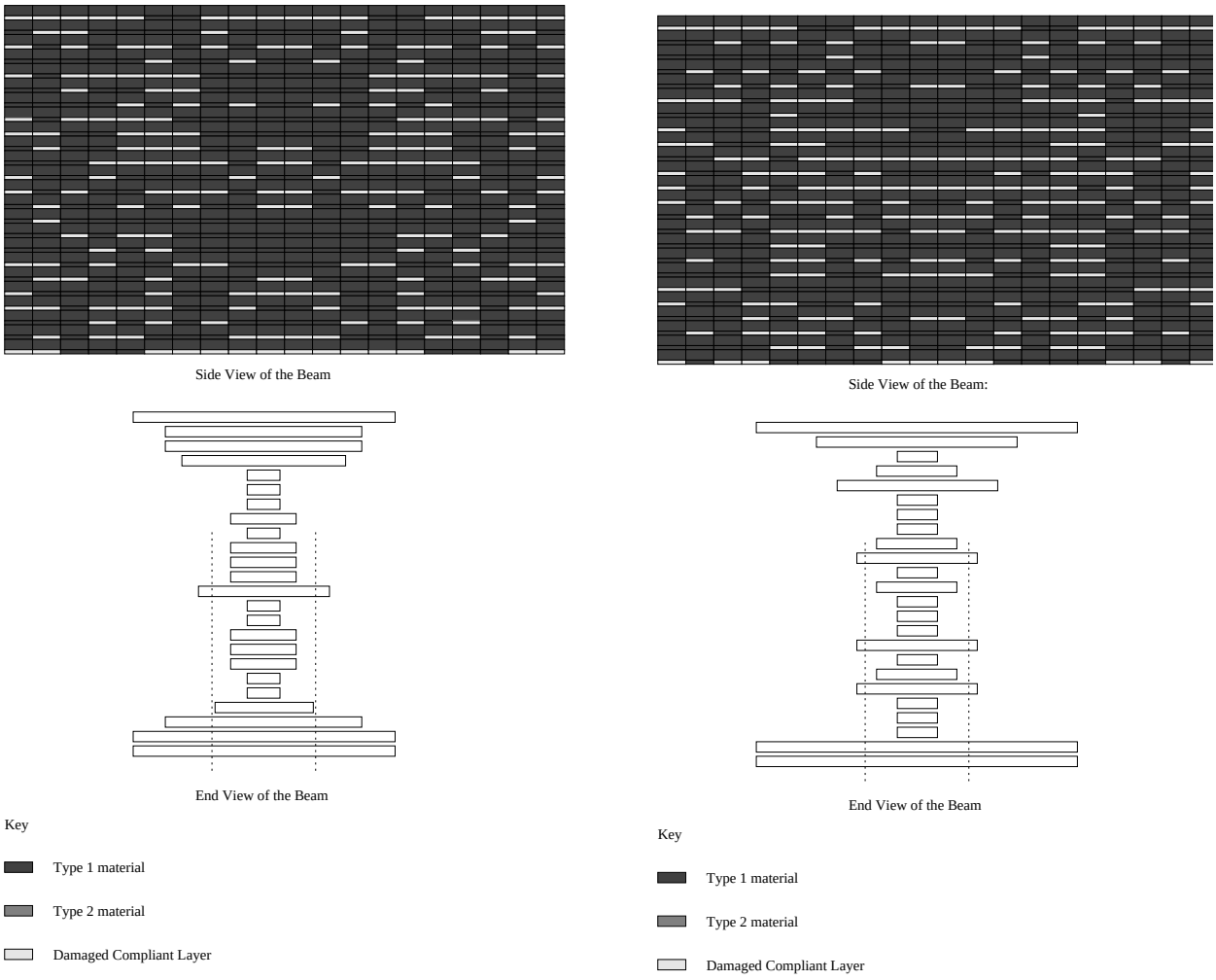

Figure 7. Two 3D beam designs obtained at different stages of the design process.

A single solution at each of two different stages of the design processing is shown in Figure 7.

that less labor is required to “layup” the laminate. One can therefore impose restrictions on the GA model concerning the composition of the layers, requiring for example that each layer must be composed of the same material. This has two benefits, namely making the resulting beam more manufacturable and also reducing the complexity of the computational problem by reducing the size of the search space. But, of course, such reduction also allows fewer potential beam structures to be considered, a disadvantage.

The representation used for our 3D beam designs is as shown in Figure 6. As before, we only directly represented half of the beam in the GA, creating the rest of the beam by mirroring that represented half across the vertical midline. We allowed only one type of material for the entire layer of the composite, but still allowed each design element individually to have the possibility of a compliant layer between it and the element below. Thus we increased the manufacturability of the beam by having only one kind of material in the layer, but allowed for compliant layers to be introduced between any two elements. The size of this part of the representation was therefore:

24 layers x (10 compliant elements per layer + 1 material per layer) = 264 bits.

We also added the 3D information not contained in the previous representations. To do so, we made the following assumptions. First, we assumed that the material and compliant element assignments made to the 2D design elements were uniform in the 3rd dimension of the design. For example, if material 1 with a compliant layer was assigned to a design element, it was assumed that the volume extension of that element in the 3rd dimension was uniform and consisted of material 1 with a compliant layer. We further assumed that the extension of every design element within a layer was the same. Thus, the width of each layer was uniform, and the only extra information that was included in our representation was the width of each layer of the beam. A constant volume constraint was imposed, with the average width of the layers denoted by the dotted vertical lines in Figure 7. We arbitrarily chose the number of potential widths to be 16 (4 bits), making the size of this part of the representation:

4 bits x 24 layers = 96 bits.

The total representation size was then:

264 bits + 96 bits = 360 bits.

6. Three-Dimensional Beam Design and Manufacturability Constraints

Our final experiments concerned taking what we had learned from the 2D beam design experiments and applying it to a more complicated 3D beam design problem. We also wanted to consider manufacturability constraints as part of the optimization criteria.

Table 3: iiGA Parallelism Results for 2D Beams

Block Size	1x1	1x2	2x2	4x2	4x4
Max Strain Energy 250,000 total evaluations	1.340e+04	1.344e+04	9.550e+03	4.816e+03	8.794e+03
Max Strain Energy 400,000 total evaluations	1.504e+04	1.225e+04	(lost this node during run)	9.637e+03	9.637e+03

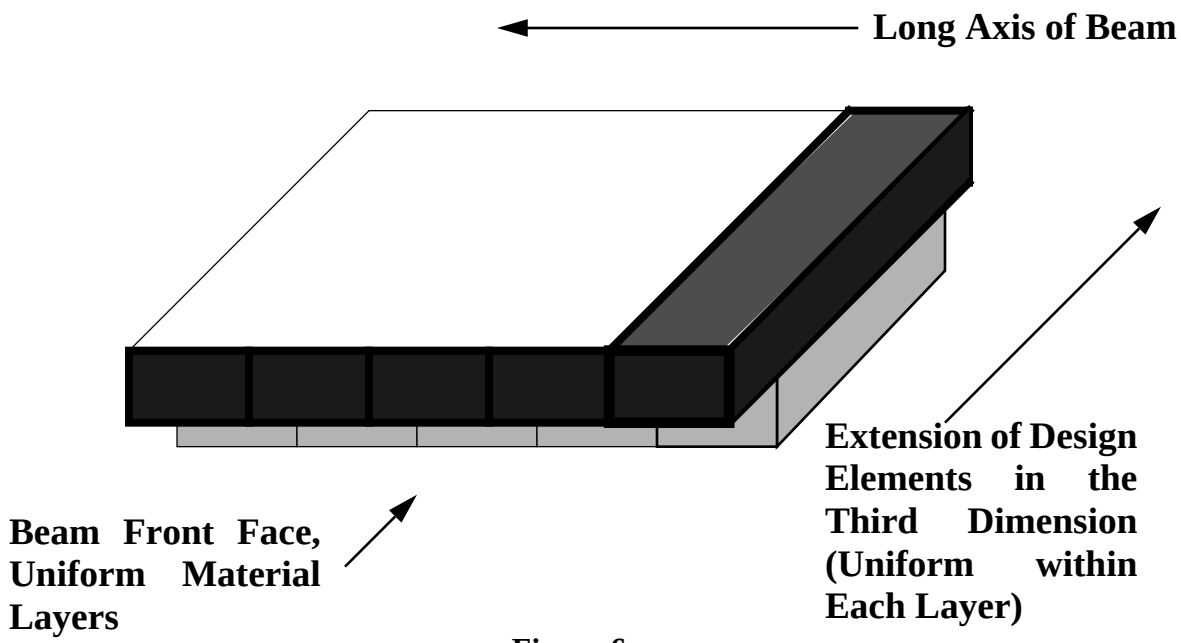

Figure 6.
3D beam composition.

Manufacturing is one of the major costs associated with using composite materials. The more exotic the material and/or layer arrangements, the more expensive is the manufacturing process. In a loose sense, the effect of design on manufacturing costs may be measured in terms of the amount of similar, consecutive elements in each layer, since an increase in such similarity means

their best individuals into the same or higher resolution nodes for “fine-grained” modification. This allows search to occur in multiple encodings, each focusing on different areas of the search space.

The main advantage an of iiGA is its search for “building blocks” in a high dimensionality search space. Each “chunkier” representation represents a sampling of the full representation, allowing faster search in the lower dimensionality space but perhaps discovering areas of interest that can be more thoroughly explored at full dimensionality.

We have applied the iiGA approach to the composite material problem in the following way. We created “chunkier” representations by representing *multiple* design elements of the original 480 elements as a single GA entry. Thus we can speak of a 1x2 chunk, which represents two elements of a row of the design as a single GA element, or a 4x4 chunk which represent a 4-row, 4-column set of elements as a single GA entry. The 1x1 chunks comprise the original 480-bit representation, and it is into this representation that each of the other representations ultimately injects its best solution.

Table 3 contains some typical results of applying iiGAs to the laminated composite beam design problem. Five different representation schemes were created as 5 separate populations on 5 processors. Every 100 generations, the best string from the 1x2, 2x2, 2x4 and 4x4 representations was converted to a 1x1 representation and injected into the 1x1 population for “fine tuning”. Note again that the exchange of information was one-way, from each of the chunkier representations into the original representation. The table reports the *total* number of evaluations for all 5 nodes, as well as the best solution fitness for each representation. Two progress reports are given, one at 250,000 total evaluations and one at 400,000 total evaluations. It is interesting to note that in 400,000 evaluations, the iiGA achieved better results than the micro-grained approach of Table 2 obtained at 1.2 million evaluations, and was still making significant progress.

In Table 3, the result from the 1x2 resolution search appears to have worsened as the search continued. This is explained as follows. When any one of the processors conducting a low resolution search converged, that processor was reinitialized and a new search was begun. After 400,000 total evaluations, the low resolution searches had all restarted at least once, so it is possible at any subsequent stage of the process that a poor “best” result will be predicted by one of these nodes.

5.2 Island Injection Parallelism

In our work with island parallelism, we experimented with a number of exchange topologies, attempting to discover which topologies would be most appropriate, if not in general, then at least for the laminated composite beam example. In so doing, we began work with a new topology we call injection island parallelism or iiGAs. The two most interesting aspects of an iiGA are its migration rules and the heterogenous nature of its nodes.

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

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

where *l* and *m* are integers, $l, m \geq 1$.

Such an encoding has the following basic properties,

(i) The smallest block size is 1*x*1. The search space is 2^n.

(ii) The largest block size is *p*x*q*. The 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. In governing interchange of individuals, we only allow a one-way exchange of information, with the direction being from a low-resolution (larger chunks) 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*

example of the kinds of results we obtained. Again, all experiments used a 480-bit representation (one half of the design represented directly, then mirrored across the vertical midline) with the max strain energy before *any* element failed as the optimization criterion.

Table 2: Results for 2D beams using parallel processing.

Parallelism Type	Population Size	Machine	Number of Evaluations	Max Strain Energy	Time
Micro-Grained, 5 Nodes	1 population of 1000	TC2000	1,200,058	1.412e+04	1d 3hr 15m
Island Parallel, 5 Nodes (ring exchange topology, exchange best every 100 generations)	5 populations of 200 each	Sparc 10	1,180,448 (≈ 236,000 per node)	2.278e+04 (best of 5)	1d 2hr 12m

A few things to note. First, micro-grained parallelism, using a slower kind of machine (TC2000 is a BBN shared memory machine using Motorola 96000 nodes) but 5 processors, reached a comparable answer to that shown in Table 1, but using considerably less time. In experiments we have conducted under more controlled conditions [24], we have found that micro-grained parallelism gives essentially linear time speedup; that is, every node added to the system contributes to a linear decrease in time required.

More important are the results of the island parallel work. In these experiments, we used 5 nodes arranged in a ring, each node working independently on a population of 200 (for a total of 1000, as in the micro-grained experiments). The best solution of each population was migrated to its neighbor around the ring every 100 generations. Note that in a comparable number of evaluations, the island parallel system found a *better* solution. Not only does island parallelism get the same linear time speed-up found in micro-grained parallelism, but also does a better job of search due to the interaction synergism of the independent populations, locating better answers in the same amount of time. Again, under more controlled experiments, we have found that island parallelism shows (algorithmic) *super* linear speed up to comparable answers [24]; that is, as more independent populations are added to the island parallel system, a more than linear decrease in time needed to reach comparable answers is obtained.

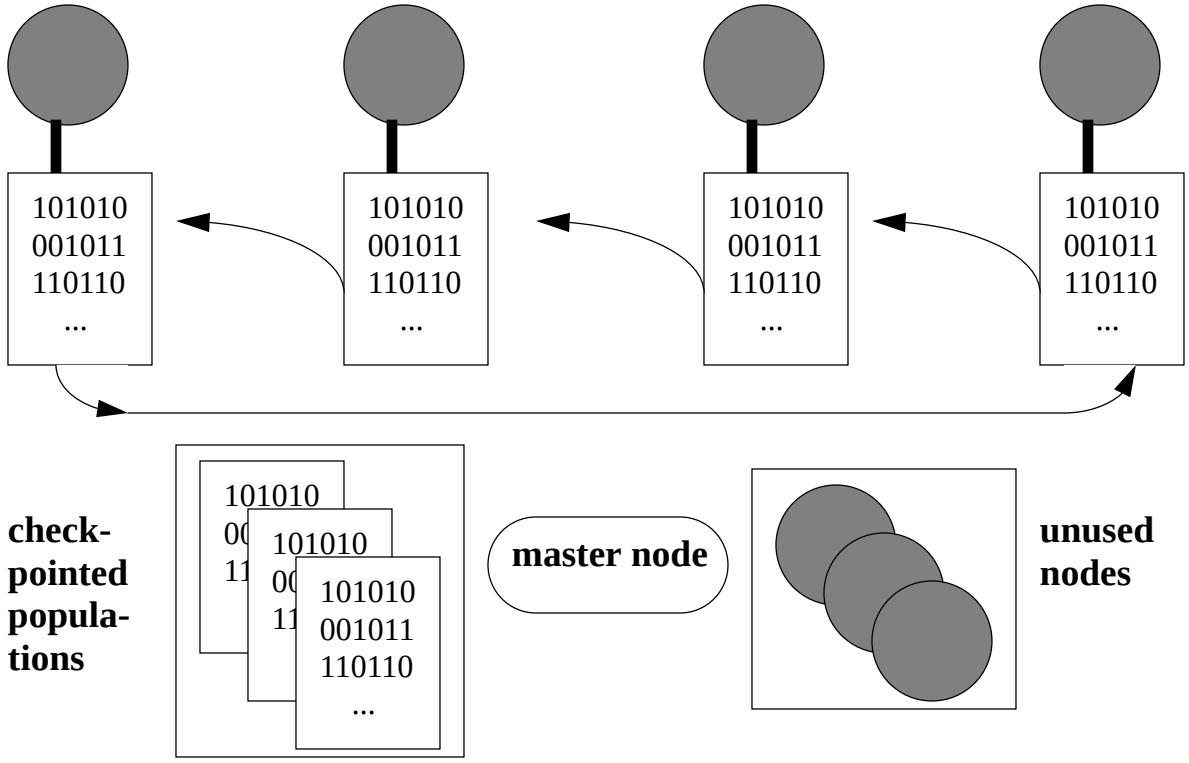

Figure 5. Island parallelism.

The island-parallel environment has also been made “fault-tolerant”. Because we run our experiments in a university laboratory environment, it is not unusual to have a processor fail (or be rebooted) in the 2-3 day period we sometimes require for solution to a difficult problem. Our environment handles such a failure by removing the lost node from the interchange topology and continuing processing with the remaining functional nodes. The loss of the population might adversely affect processing in the short term (both accuracy and time), but it does not stop the run, and the process can typically recover from this. Furthermore, if other nodes are available, the cached population can be migrated to a free processor and the population reintroduced into the interchange topology. Our environment is capable of this as well, though it tends to be impractical in the university lab setting where machines are nearly always busy. We can also run in a “polite” mode in public laboratories, in which case our populations “go to sleep” whenever a user sits down at a workstation, resuming automatically only after an idle period is detected. Again, the temporary loss of a processor in this way slows the overall process somewhat, but the optimization continues, nonetheless.

The effectiveness of these two parallel architectures is shown in Table 2, which is a typical

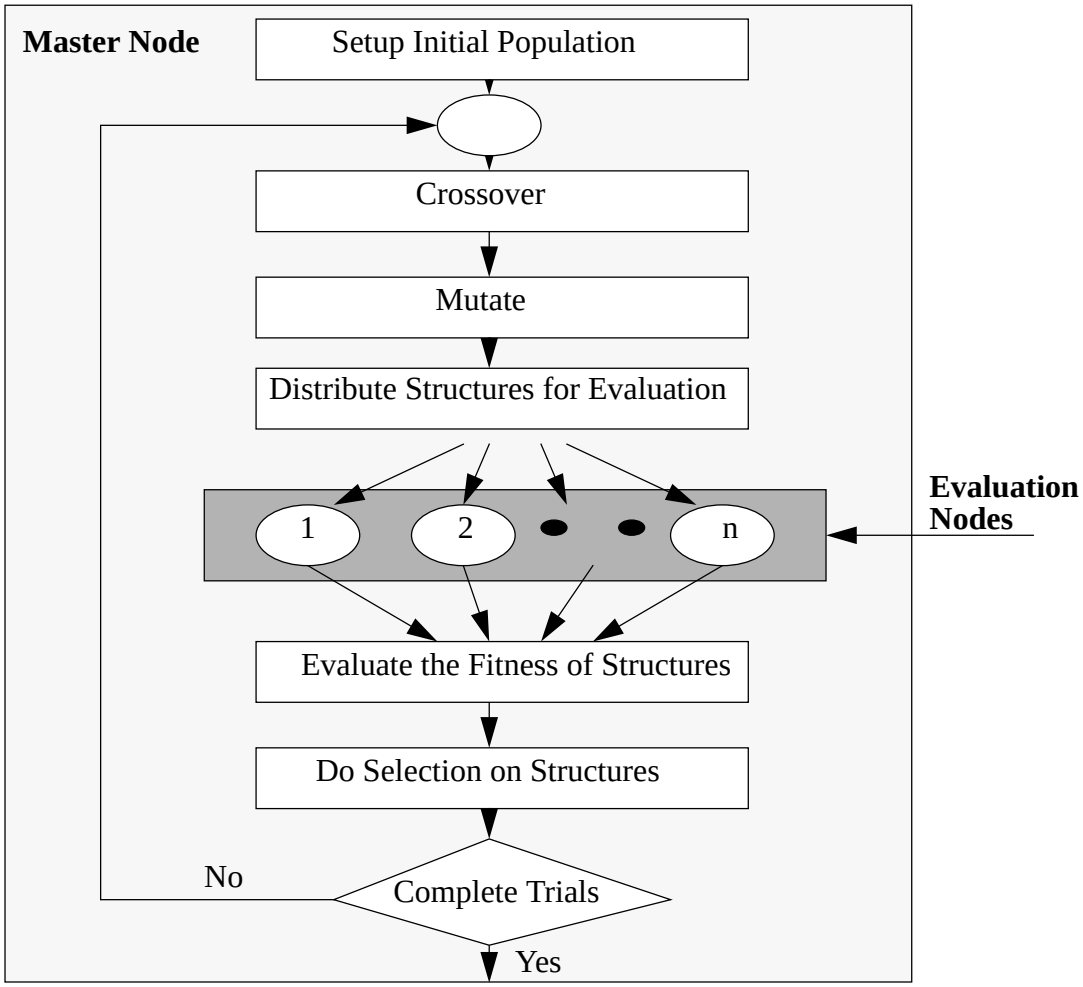

Figure 4. Micro-grain parallelism.

The other mode of operation, which is even more fruitful, maintains multiple, separate populations on each processor (see Figure 5). These separate populations explore their own areas of the search space, but occasionally interchange their best (typically) solutions. The result of this "island parallelism" is that the separate populations search different parts of the solution space relatively independently; but, based on "suggestions" from other populations, begin to converge and devote their efforts in parallel to promising areas. This approach does indeed decrease the number of total evaluations needed to find optimal solutions, and is less likely to get stuck at a local optimum than is a single population, given similar parameter settings.

bit representation of the beam, creating a search space of 2^{960}. This design run required approximately $8x10^6$ design evaluations. The result does not show midline symmetry, nor any other obvious form of symmetry or recognizable pattern. It was more computationally expensive than the representations such as the first "shift" representation, but gave better results.

As a result of these experiments, the representation of choice used for the remainder of our experiments was 480 bits long, representing half of the beam. The remainder of the beam design was generated by mirroring the 480-bit solution across the vertical midline. The results we obtained from this representation were always comparable to the few 960-bit experiments we performed and much less expensive computationally than the full 960-bit representation.

5. Parallel Distributed Processing for GA's and Design

5.1 Parallel GA Approaches

We have previously successfully applied GA's to data mining applications [28]. As part of that work we developed a parallel processing environment for Genetic Algorithms based on the p4 parallel processing tool [5]. The environment runs on both shared memory parallel processors and on distributed networks of workstations. This environment runs in two modes. The first mode assigns a processor to every solution in the population, termed "micro-grain" parallelism (see Figure 4). This processor is used to do the evaluation of the GA solution for the particular application. Its major advantage is one of time. If a processor is available for every "solution" in the current population, then the time to evaluate the fitness of the entire population is reduced to little more than the time needed to evaluate a single solution (communication overhead is minuscule relative to the processing time for evaluation of a single solution). This approach can be useful if the evaluation function is very expensive computationally; however, unlike the techniques described below, it does not decrease the total number of evaluations needed to find the best solution, and it does require essentially synchronous operation of the multiple processors

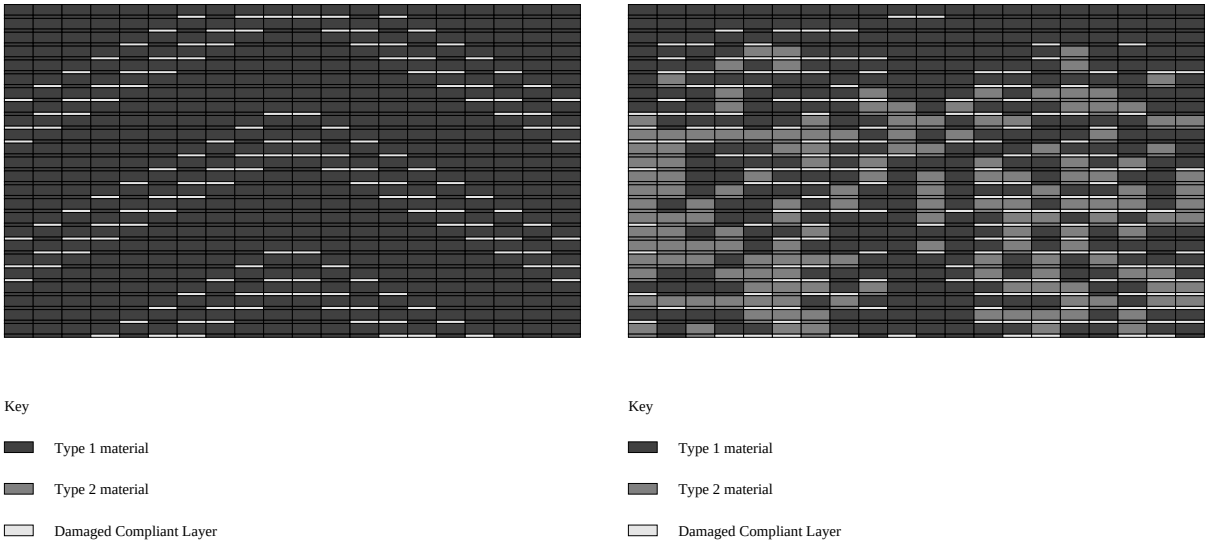

Figure 3. Two examples of beam designs using different representations.

Table 1: Some 2D Beam Results

Experiment Representation Type	Machine Type	Population Size	Number of Evaluations	Max Strain Energy	Time
960-Bit (run 2)	Hp 735	1000	3,990,094	1.077e+04	≈ 4 days
10-Bit	Sparc 10	100	10,000	7.532e+03 (optimal)	≈ 10 min.

The first design (Figure 3, left) used a simple representation of only 10 design elements (only 10 elements were actually represented in the GA solution string). These 10 elements were used to "generate" a full design, which was then evaluated. The 10 design elements constituted the left half of the first layer. The subsequent lower layers of the left half of the design were generated using the 10-element layer above. Each layer (except for the first one, generated by the GA itself) was generated by a "left shift" of one element of the layer above, resulting in 240 design elements in the left half of the design. This half design was then "mirrored" across the vertical midline generating the remaining 240 elements of the design and forcing a midline symmetric design. This solution space was quite small (approximately 10^6 designs), and therefore enumerable. We confirmed that the design found by the GA was indeed the *optimal* design under the constraints of the representation. Note that despite the availability of four materials, only two were used throughout this design, making it much more manufacturable.

The second, less symmetric design (Figure 3, right), was the result of a run using the full 960-

ditions of continuous displacement and transverse shear stress at each layer interface as well as by satisfying the shear traction conditions on the top and bottom faces of the laminate. The width and thickness of each layer is arbitrary, so that beams of various cross-sectional shape may be analyzed. The resulting model is accurate for thin and moderately thick laminates with material properties that vary arbitrarily through the thickness, yet is efficient enough for optimal design calculations.

3.3 Implementation

For our GA experiments we used a modified version of GAucsd [30], a public domain C implementation Genetic Algorithm tool. Our modifications were chiefly the addition of distributed parallel processing code from p4 [5], a tool for building parallel systems, that enabled us to do GA's over a network of Sun workstations. This code has been used in a number of other domains by the MSU GA group [24, 28]. The laminate analysis code of Averill [3] was written in Fortran and linked into the GAucsd code as the evaluation function.

4. Energy-Absorbing Beam Design

4.1 Simple 2D Beam results

Our first experiments focused on various ways to represent a composite structure in a GA string and the effects of those representations. All of these experiments were done using a single population.Two example designs, representing a range of complexity of GA representations, are shown in Figure 3, along with the details of their runs in Table 1. In the table, "evaluations" represents the number of individual solution evaluations performed. Max Strain Energy is the measure of energy absorption used (see Section 3.1), and is the optimization criterion.

Table 1: Some 2D Beam Results

Experiment Representation Type	Machine Type	Population Size	Number of Evaluations	Max Strain Energy	Time
960-Bit (run 1)	Hp 735	1000	7,910,349	1.891e+04	9d 9hr 1m

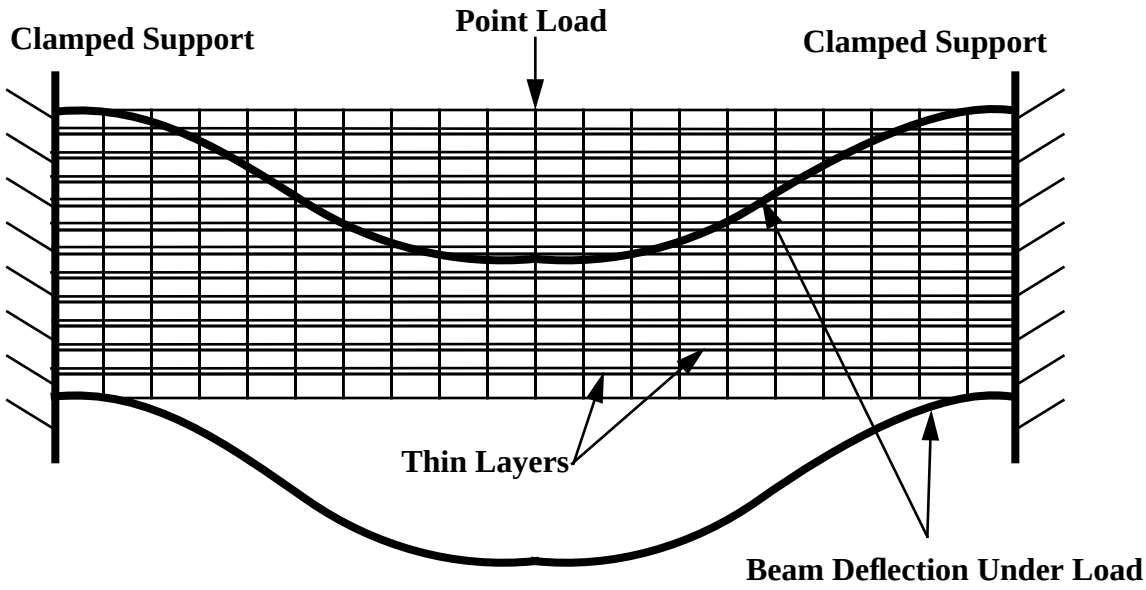

Figure 2. Clamped-clamped laminated beam with center point load.

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 maximum stress 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.

3.2 Laminate Structural Analysis

In recent years, a great deal of research has been devoted to the development of laminate theories that account for the most prominent geometric and structural features on a layerwise basis, while minimizing the number of parameters (or degrees of freedom) required to describe the structural response (see, e.g. [2, 3, 11]). In the current study, the laminated beam model developed by Averill and Yip [3] is employed to analyze each possible design. This model accurately accounts for the layerwise variations of displacements and stresses in laminated composites by assuming a piecewise continuous through-the-thickness distribution of the inplane displacement. The number of degrees of freedom required to describe the laminate behavior is reduced by imposing the con-

The eventual goal of our research group is to develop tools for designing 3D large scale laminated composite structures 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 initial studies presented here focus on the design of laminated composite beams. 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 (see Figure 2). 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.

In particular, a 24-layer beam made of graphite-epoxy composite layers is considered with clamped-clamped end conditions and an applied point load at midspan (see Figure 2, where only 12 structural layers are shown). A thin layer (about 5.5% of the nominal ply thickness) is placed at the top of each composite layer, so there are actually 48 layers in the model. Each thin layer may be assigned the same material properties as the layer immediately below it, or it may be assumed that the thin layer is compliant, with stiffness properties three orders of magnitude less than the composite layers. The length-to-thickness ratio of the beam is 50. The length of the beam is divided into 20 sublengths (finite elements), so there are 20x48 = 960 design elements. The GA must decide whether to place a 0 degree ply (henceforth called material 1) or a 90 degree ply (henceforth called material 2) in each of the 480 structural ply design elements, and whether or not to place a compliant material in each of the 480 thin layer design elements. Note that this problem is of considerably higher complexity than the other composite structure design work cited above.

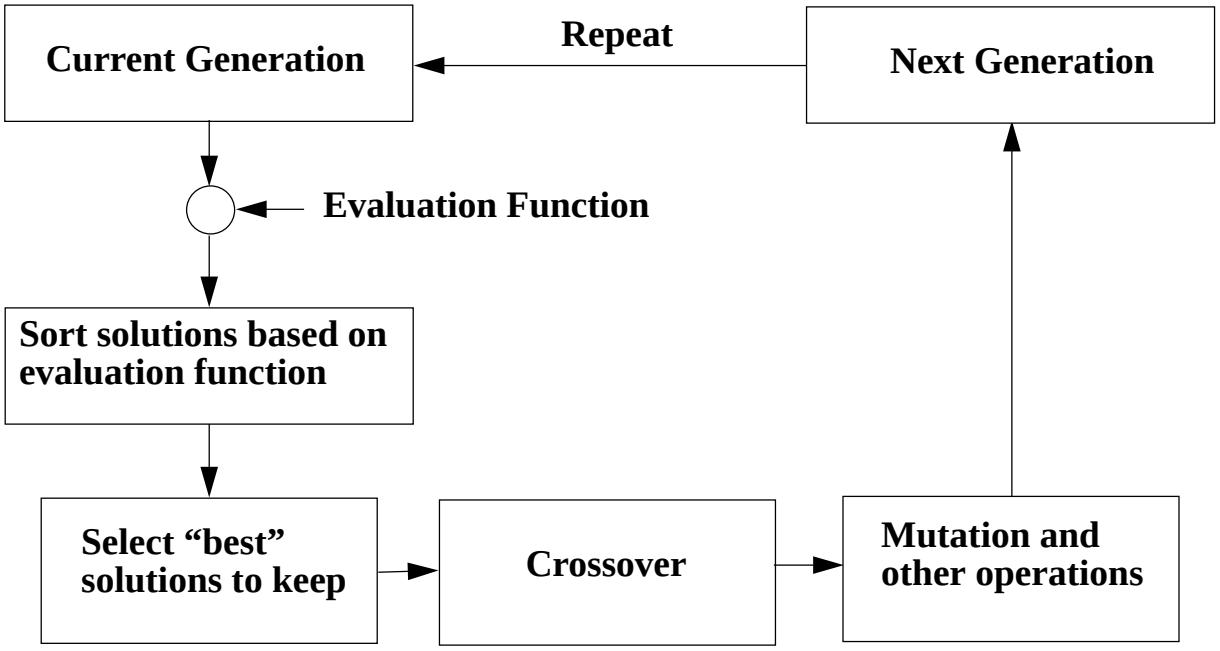

Figure 1. The basic steps in a Genetic Algorithm.

3. GA Design of Laminated Composite Structures

3.1 Background and Problem Statement

Multi-layer construction of panels offers many opportunities for analysts and designers to optimize structures for a particular or even for multiple tasks, but this flexibility results in a very large number of often discrete design variables (e.g., the thickness, material type, and orientation of each layer, as well as the number of layers). In addition, the design space may contain many local extrema, even singular extrema, and there may be many possible designs that meet or very nearly meet the design criteria. Genetic Algorithms are well-suited for this type of design problem, and efforts to apply GA's to laminated structures have been increasing rapidly over the past five years.

There are a number of groups working on the application of Genetic Algorithms to the design of composite structures. In particular, Haftka's group at Virginia Polytechnic Institute and State University have been actively pursuing GA applications to the design of laminated composite panels [21,22]. They have examined GA's using so called "memory", essentially caching previous designs and their evaluations to avoid the expense of re-evaluation [21]. Other active groups include Hajela at Rensselaer Polytechnic [15,16] and Leung and Nevill [23].

digm in many areas. It has been used in engineering applications such as clustering [12] and pipeline optimization [13]. In the context of design, GA's have been used for: VLSI cell placement [20], floor plan design [26], air-injected hydrocyclone [18], network design [10], and others.

The classes of problems encountered in design include many that are difficult to solve in practice -- i.e., NP-hard and some NP-complete problems. The computation of a *truly* optimal solution to any design problem is usually possible only for a very limited domain. Some heuristic methods must typically be applied to reduce the search space and generate sets of approximate (near-optimal) solutions. In the genetic algorithm approach, each point in the search space is called a "chromosome" or string [13], and represents a possible solution to the problem. A GA approach requires a population of chromosomes (strings) representing a combination of features from the set of features, and requires a cost function (called an evaluation or fitness function) F(n). This function calculates the fitness of each chromosome. The algorithm manipulates a finite set (population) of chromosomes, based loosely on the mechanism of natural evolution. In each generation, chromosomes are subjected to certain operators, such as crossover, inversion, and mutation, analogous to processes which occur in natural reproduction. The crossover of two chromosomes produces a pair of offspring chromosomes which are syntheses or combinations of the traits of their parents. Inversion in a chromosome produces a mirror-image reflection of a contiguous subset of the features on the chromosome. A mutation on a chromosome produces a nearly identical chromosome with only local alterations of some regions of the chromosome. A great deal of information regarding these operators is available in the literature cited above, and it will not be presented in detail here.

2.2 Operation of the GA

The optimization process is performed in cycles called generations. Figure 1 gives an overview of a typical GA algorithm. During each generation, a set of new chromosomes is created using the crossover, inversion, mutation, and crossmutation operators. Since the population size is finite, only the best chromosomes are allowed to survive to the next cycle of reproduction. The crossover rate often assumes quite high values (on the order of 80-85%), while the mutation rate is small (typically 1-15%) for efficient search. The cycle repeats until the population "converges"; that is, the diversity of the feature values among the population is very low and further exploration seems pointless, or until the answer is "good enough."

Our approach is loosely based on a generate-and-test system, using the concepts of simulation and optimization to control search. In our approach, a generation technique generates a set of complete designs. These designs are then tested/evaluated based on a simulation of the design. The optimizer feeds the simulator's performance information back to the generator, and that feedback influences the generation of additional designs based on the existing designs and their performances. This feedback controls how the space is searched, focusing on promising design features and how they can be combined into an overall "good" design. We incorporate these tools -- simulation and optimization -- as part of an overall generate-and-test strategy, in our use of Genetic Algorithms (GAs) for design.

We will show the effectiveness of GAs for design problems using a real-world test domain, the design of composite material beams, in particular beams that are optimized for energy absorption (Section 3). The paper will discuss the design of both two and three dimensional beams using a GA (Sections 4 and 6). We will also show that parallel processing techniques make GAs effective for difficult, real-world design problems such as the composite-beam design problem. In particular we introduce a refined parallel-processing GA called the injection island GA (iiGA). iiGAs use multiple representations of the same problem to explore different design spaces simultaneously, then combine the results of those good designs into an even better design (Section 5). We will then discuss the significance of the work and our future plans (Section 7).

2. Design via Genetic Algorithms

2.1 Overview of Genetic Algorithms.

In 1975 Holland [17] described a methodology for studying natural adaptive systems and designing artificial adaptive systems. It is now frequently used as an optimization method. The biological basis for this adaptation process is Darwinian natural selection and Mendelian genetics, that is elimination of weak elements by favoring retention of optimal and near-optimal individuals("survival of the fittest") and recombination of features of good individuals to perhaps make better individuals. References [13,17] contain a theoretical analysis of a class of adaptive systems in which the search space of the problem, in our case the space of structural modifications of composite beams, is represented by sequences (strings) of symbols chosen from some alphabet (usually a binary alphabet). The searching of this representation space is performed using so-called genetic algorithms. The genetic algorithm is now widely recognized as an effective search para-

Design Using Genetic Algorithms -- Some Results for Laminated Composite Structures.

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

Abstract

This paper reviews an approach to design using Genetic Algorithms as applied to laminated composite structures. We first discuss the concept of a Genetic Algorithm (GA) and the role it can play in design, namely as an *evolutionary search optimizer* based on *simulation results*. We then discuss the composite structure design problem and how GA's can be used for design in this domain. Finally we discuss various experiments we have conducted and our results in using GA's for the design of energy absorbing composite material structures, including: 2D beam design, 3D beam design, use of distributed processing to increase efficiency and accuracy of GA's in design, and inclusion of manufacturability constraints in automated design.

1. Introduction

Design is an open-ended problem-solving area, encompassing a wide range of problem-solving types and approaches. It is therefore difficult to reduce all descriptions of the various design approaches to a common model, but Coyne et. al. [9], in describing design as a state space search [27], make the following statement:

> Problem solving implies search. What methods are available to control search? The two most prominent computer-aided design techniques employed in producing "good" designs are simulation and optimization. *Simulation* predicts the performance of a given potential design, while *optimization* identifies potential designs for a given performance goal. (pg. 17)

Many design problem-solvers make use of both simulation and optimization to generate designs. In fact, one way to view this coupling is under the general problem-solving technique called *generate-and-tes*t, which has been used in a number of design and design-like systems [4, 26]. The problem encountered in the use of generate-and-test systems is control of search. Blind generate-and-test systems might enumeratively explore each possible design, an impossibility given a design of any reasonable complexity. Thus control of search is paramount in such systems.

Design Using Genetic Algorithms -- Some Results for Laminated Composite Structures.

William F. Punch[1], Ronald C. Averill[2], Erik D. Goodman[3],
Shyh-Chang Lin[3], Ying Ding[1]

[1]Intelligent Systems Laboratory, A714 Wells Hall, Computer Science Department, Michigan State University, punch@cps.msu.edu. All correspondence should be addressed to Punch.

[2]Materials Science and Mechanics Department, Michigan State University, averill@egr.msu.edu

[3]Case Center for Computer-Aided Engineering and Manufacturing, College of Engineering, Michigan State University, goodman@egr.msu.edu

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

Abstract

This paper reviews an approach to design using Genetic Algorithms as applied to laminated composite structures. We first discuss the concept of a Genetic Algorithm (GA) and the role it can play in design, namely as an *evolutionary search optimizer* based on *simulation results*. We then discuss the composite structure design problem and how GA's can be used for design in this domain. Finally we discuss various experiments we have conducted and our results in using GA's for the design of energy absorbing composite material structures, including: 2D beam design, 3D beam design, use of distributed processing to increase efficiency and accuracy of GA's in design, and inclusion of manufacturability constraints in automated design.

Accepted to IEEE Expert, Jan 1995.