

**Optimization Frameworks for Discrete  
Composite Laminate Stacking Sequences**

by

David B. Adams

Dissertation submitted to the Faculty of the  
Virginia Polytechnic Institute and State University  
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science and Applications

APPROVED:

---

Layne T. Watson

---

Zafer Gürdal

---

Calvin J. Ribbens

---

Lenwood S. Heath

---

Christine M. Anderson-Cook

July 20, 2005

Blacksburg, Virginia

**Key words:** Composite Laminates, Genetic Algorithms, Parallel Computing, Combinatorial Optimization, Decomposition, Blending.

# **Optimization Frameworks for Discrete Composite Laminate Stacking Sequences**

by

David B. Adams

(ABSTRACT)

Composite panel structure optimization is commonly decomposed into panel optimization subproblems, with specified local loads, resulting in manufacturing incompatibilities between adjacent panel designs. Using genetic algorithms to optimize local panel stacking sequences allows panel populations of stacking sequences to evolve in parallel and send migrants to adjacent panels, so as to blend the local panel designs globally. The blending process is accomplished using the edit distance between individuals of a population and the set of migrants from adjacent panels. The objective function evaluating the fitness of designs is modified according to the severity of mismatches detected between neighboring populations. This lays the ground work for natural evolution to a blended global solution without leaving the paradigm of genetic algorithms. An additional method applied here for constructing globally blended panel designs uses a parallel decomposition antithetical to that of earlier work. Rather than performing concurrent panel genetic optimizations, a single genetic optimization is conducted for the entire structure with the parallelism solely within the fitness evaluations. A guide based genetic algorithm approach is introduced to exclusively generate and evaluate valid globally blended designs, utilizing a simple master-slave parallel implementation, implicitly reducing the size of the problem design space and increasing the quality of discovered local optima.

## ACKNOWLEDGEMENTS

I would like to thank Prof. Layne T. Watson for serving as my research advisor and committee chairperson. His broad perspective and vision coupled with his tireless work ethic create in him the paragon of what a research advisor should be. He's seen me through to the end of an era and helped place me on the road to the next. I can not thank him enough for his encouraging words and understanding stance during these last few months of my life. I wish to also thank Zafer Gürdal and Omprakash Seresta for providing me with the engineering reviews required for the work.

I am indebted to the Terascale Computing Facility at Virginia Tech for a generous grant of computer time on System X as well as time on the Anantham cluster.

To my friend D. Slotta who in times of great need always seemed to be there to lend a helping hand and remind me to keep things in perspective, I will not forget.

To P. Haggerty who graciously let me live in his home during the completion of this work while I was homeless, I am indebted forever.

Finally, to Elissa, thanks for the years we had together. I regret that you are not present for the completion of this work but I will never forget you. Take care.

## TABLE OF CONTENTS.

1. Introduction .....	1
2. Genetic Algorithms.....	4
3. The Blending Problem .....	6
3.1. Laminate Encoding .....	7
3.2. Selection Strategies .....	9
3.3. Serial GA Pseudo Code .....	11
3.4. Edit Distance .....	11
3.5. Parallel GA Pseudo Code .....	14
4. Reference Migration Results .....	18
4.1. Ring Communication Topology .....	19
4.2. Physically Matched Communication Topology .....	21
4.3. Weight-Cost Tradeoffs .....	24
4.4. Umbra Migration .....	27
5. Guide Based Design .....	31
5.1. Master-Slave Parallelism .....	34
6. Guide Based Design Results .....	36
7. Conclusions for Fixed Load Methods .....	39
8. Coupled Load Guide Based Design Extension .....	41
8.1. Guide Based Design .....	42
8.2. Guide Design Subproblem .....	44
8.2.1. Global/Local Iteration .....	45
8.2.2. Middle Tier Addition .....	50
8.3. Results .....	53

8.3.1. Simple Example .....	53
8.3.2. Load Balancing Mechanism .....	55
8.3.3. Comparison with Other Published Work .....	56
9. Conclusions .....	61
REFERENCES .....	63
VITA .....	65

**LIST OF FIGURES.**

Figure 1. Local Optimization Problem ..... 6

Figure 2. Eighteen Panel Irregular Grid ..... 18

Figure 3. Migration Box Example ..... 22

Figure 4. False Optimum of Topology Mapped Communication ..... 23

Figure 5. Weight-Cost Tradeoffs ..... 25

Figure 6. Edit Distance Pair Plot ..... 28

Figure 7. Guide Based Optimization Example ..... 33

Figure 8. Outwardly/Inwardly Blended Distinction Example ..... 43

Figure 9. Guide Graph Example ..... 48

Figure 10. Top Tier Flow Chart ..... 49

Figure 11. Middle Tier Flow Chart ..... 52

Figure 12. Eight Panel Wing Structure Model ..... 54

Figure 13. Eight Panel Layout ..... 54

Figure 14. Eighteen Panel Wing Structure Model ..... 57

Figure 15. Eighteen Panel Layout ..... 57

Figure 16. Topology Mapped Panel Thickness ..... 60

**LIST OF TABLES.**

Table 1. Static GA Parameters .....19

Table 2. Control Run ..... 20

Table 3. Ring Topology Blending(1) .....22

Table 4. Ring Topology Blending(2) .....26

Table 5. Ring Topology Blending(3) .....27

Table 6. Ring Topology Blending(4) .....30

Table 7. Static GA Parameters(2) .....36

Table 8. Guide Based Design ..... 37

Table 9. Other Guide Based Designs ..... 38

Table 10. Stacking Sequence of Top Panels .....59

Table 11. Stacking Sequence of Bottom Panels ..... 59

Table 12. Panel Thickness and Mass ..... 60

## Chapter 1: INTRODUCTION

The design of optimal composite laminates has been shown to be well suited to the defining characteristics of genetic algorithms. Techniques for improving the efficiency of this methodology have been explored for several problems using local improvement, memory, migration, and varied selection schemes [20], [14], [29]. For large structures, such as the design of a wing or fuselage, the optimization is divided into smaller, tractable, subproblems using predefined local loads to constrain the optimization [20], [2], [30], [15]. The design process requires the specification of the stacking sequence, which is defined by the orientation and material type of each ply layer, creating a discrete optimization problem. It is computationally expensive to design an entire wing or fuselage structure with the panels optimized simultaneously. Instead, local panels are commonly optimized for the specified local loads by ignoring the possible continuity of some or all of the layers from one panel to another across the structure. The stacking sequences of the final panel designs in this scenario are usually incompatible across the overall structure due to the widely varying design possibilities for optimal and near optimal stacking sequences.

The primary objective is, for specified local panel loads, to design the individual panels such that continuity of the ply orientations across laminates is achieved while minimizing the total structural weight. Soremekun et al.[29] introduced multiple elitist selection schemes that by nature aid in discovering alternative designs with similar fitness values. In a standard elitist selection strategy, only a single member of a parent population can survive the selection process without being modified and be placed in the child population. In a multiple elitist selection strategy the genetic algorithm allows a greater number of high fitness members to survive the selection process at each generation. Building on the fact that a local panel optimum (or near optimum) design can correspond to several distinct stacking sequences, perhaps a globally blended solution, conforming with high fitness designs of neighboring panels, can be derived with intelligent selection from a large spectrum of alternative elite designs of a population.

The first portion of this work focuses on using genetic algorithms in parallel to achieve compatible local panels. Instead of optimizing local panels in isolation, each individual



panel is designed by a different processor in a parallel environment. The populations representing each panel evolve in parallel and periodically send migrant individuals to adjacent populations. This migration is similar to the work presented by McMahon and Watson [20], but differs in the fact that migrant individuals do not have to enter the population once they arrive at their destination. Instead, migrant individuals are stored and referred to as needed to assess the compatibility of current evolutionary trends with neighboring populations, reserving the ability to induct migrants. A metric is applied, in this case the string edit distance, to evaluate the similarities in the current population with members stored as migrants. The computed distance value is used to reward (increase the fitness of) those individuals in a population that are evolving closely to those found in the migrant populations from adjacent panels. This rewarding process is a small modification to the objective function fitness value and serves to blend the panel designs of the population, thereby balancing the optimization of weight with stacking sequence compatibility.

Work on genetic algorithm (GA) based blending includes the use of the edit distance metric to allow a set of independently evolving panel populations to evolve to a blended global solution using reference migration [2], the use of sublaminates definitions and design variable zones by Sormekun et al. [30], and the addition of continuity constraints proposed by Liu and Haftka [17]. The edit distance method utilizes a multiple-population parallel approach that obtains blended designs through evolutionary pressures from neighboring populations. It is a nonstandard parallel decomposition that works asynchronously in real time to emulate semi-isolated populations with random migration to produce blended global designs from a pool of globally unconstrained local stacking sequence design possibilities. Evolutionary pressures are controlled through a user defined scaling factor that modifies the severity of penalties imposed for blending mismatches. These penalties, however, were found to hinder convergence to a global optimum by creating local optima in the search space that are artifacts of the algorithm itself [2]. The approach used in [30] is a two step procedure that relies on first optimizing the individual panels followed by identifying common thickness zones across multiple panels that are redefined and reoptimized using blended stacking sequences. Because of the heuristic nature of the approach, however, it

is possible that suboptimal designs are generated, although blended designs are obtained with little weight penalty compared to unblended minimum weight structural design.

The second portion of this work focuses on the introduction of a guide based genetic algorithm that implicitly reduces the search space of possible designs by forcing the generation of global (defining the entire structure) individuals that are always completely blended. Multiple populations are replaced by a single population of interbreeding individuals, concentrating the parallelism in master-slave load distribution analysis. This method is a fundamentally different parallel decomposition of the problem from that in Adams et al. [2]. Instead of discovering a globally blended design at the end of the design process, all designs considered are always blended throughout the genetic algorithm. The final portion of this work extends the idea of guide based design into a more realistic representation of the problem in which the local loads are affected by changes in the stacking sequence of other panels.

Chapter 2 provides some background on the work in composite laminate design using genetic algorithms. Chapter 3 introduces the reference migration blending algorithm providing pseudocode for a serial genetic algorithm and the direct parallel implementation. Chapter 4 presents results for an 18 panel design problem in multiple communication topologies with varying parameters applied to the blending process. Chapter 5 defines the guide based genetic algorithm approach and outlines the parallel distribution of work. Chapter 6 presents results for the 18 panel design problem with comparisons to reference migration. Chapter 7 gives some concluding remarks about fixed load blending methods and Chapter 8 presents the guide based design work extended to incorporate local loads that are coupled with the stacking sequence for all panels.

## Chapter 2: GENETIC ALGORITHMS

A genetic algorithm (GA) is a directed search algorithm using ideas based on natural selection to guide the exploration of the search space toward a global optimum. Common elements that occur in most genetic algorithms are those of population initialization, parent selection, crossover, mutation, and the selection of successive generations. Population initialization here is a random generation of all initial starting population features including ply orientations and ply thicknesses. The process of parent selection is the method by which members of the population are chosen for reproduction. Chapter 3.2 details the roulette wheel method used in this work. Crossover is a standard genetic operator that emulates the combination process of taking two parents and creating a unique child. Additional details on a one point crossover method used in this work can be found in Chapter 3.2. Mutation in genetic algorithms introduces random elements of change into child populations allowing for the possibility of a single ply layer to mutate its orientation after crossover. The probability of mutation represents the chance that a single gene will change orientation while the location of the mutated gene is also selected at random. Each element of the algorithm has many variations, modified to suit the needs of the problem at hand, including but not limited to attempts to mimic natural genetics in every phase. The original work on genetic algorithms is attributed to Holland [11] in 1975, with application work following soon after in static function optimization by DeJong [12]. Goldberg [8] popularized the idea with his book in 1989, and is cited extensively in the literature defining genetic algorithms as search procedures based on the mechanics of natural selection and natural genetics. Bäck [5] is an excellent recent reference. Much of the work today using genetic algorithms can still fit this definition, though the concepts of natural selection and genetics are expanded to encompass some unnatural elements beyond the pristine translations from biology. In practice, genetic operators are tailored to specific problems in ways that have no analog in nature [20], [14], [24].

The work done in designing composite laminates by GA has spawned some genetic operators that work well for this type of problem, but are not in the scope of the standard genetic algorithm operators (crossover and mutation). Le Riche [24] introduced the concept

of a two-stack swap operator, which is generalized in the permutation and swap operators of McMahan [19]. McMahan also describes the addition and deletion operators specific to composite laminate designs with varying numbers of plies.

All of the operators just mentioned and a multiple elitist strategy introduced by Soremekun et al.[29] are used in this work. Brief descriptions follow. The addition and deletion operators can randomly add or delete a gene from the chromosome string. Though the encoding of the chromosome remains at a fixed size, the number of active genes in the chromosome can fluctuate randomly through additions and deletions with empty gene positions represented by a zero encoding. The permutation operator inverts a randomly determined sequence of genes (random in both length and position of the string) and lastly, for the new operators, the swap operator randomly switches the positions of two active genes in a chromosome. The multiple elitist strategy employed in this work is simply a varied selection scheme based on common elitist selection. Multiple elitist selection as described here refers to the number of members of the parent population that are not destroyed, but rather are carried on into subsequent generations, replacing the weakest individuals of the generated child population. This multiple elitist scheme is designed to preserve more information from a previous generation than allowed by the elitist selection, while keeping the number of surviving elite at a constant level throughout the computation. Examples of these operators and multiple elitist selection are given in the next chapter.

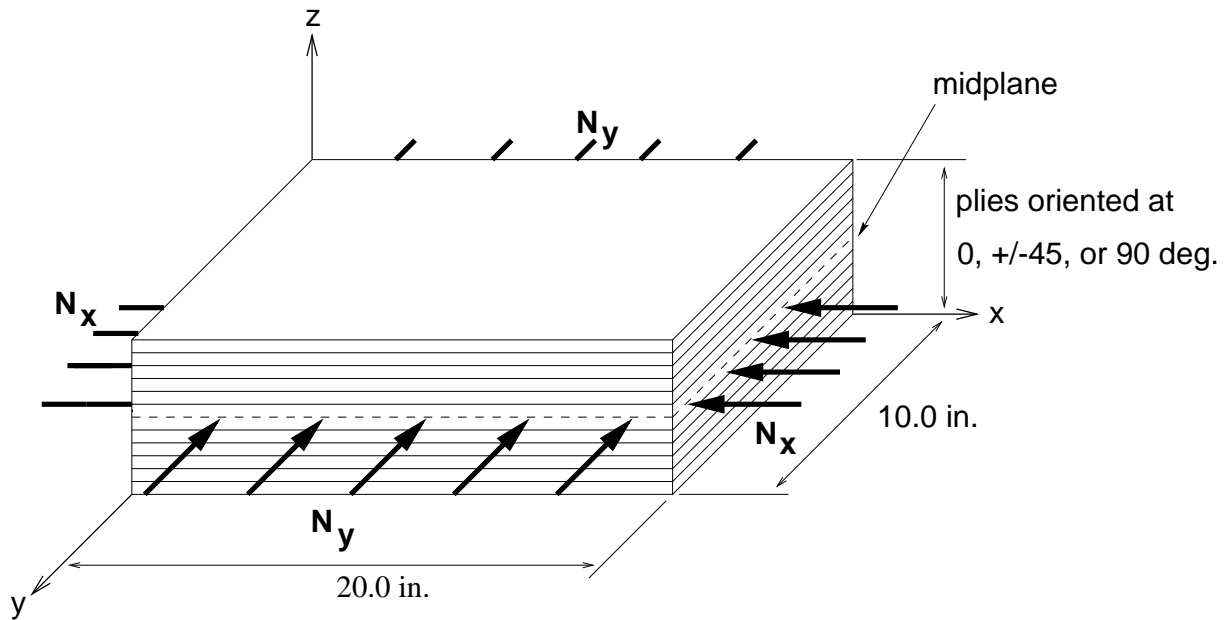


FIG. 1. *Local optimization problem for laminate stacking sequence.*

### Chapter 3: THE BLENDING PROBLEM

The local design problem presented here is a laminate stacking sequence optimization where the goal is to find the number of plies with material type and orientation that yield the best performance for a given set of loading conditions. The laminate is assumed to be balanced, symmetric, and constructed with a finite number of possible fixed orientations for each ply. Figure 1 shows an example of such a laminate. These simplifications aid in the encoding of the laminate and are common in the literature [20], [14], [29]. In the global design problem, a single panel optimization represents only a fraction of the entire design. The global solution created from these seemingly unconnected components is, in practice, infeasible to manufacture due to the widely varying design possibilities for optimal and near optimal stacking sequences. The approach described in the following sections is one of several attempts to blend locally optimized composite panels into a global design including the addition of heuristic continuity constraints proposed by Liu and Haftka [17].

The blending problem here is defined as a laminate stacking sequence optimization on multiple interconnected panels with given local loading constraints. The goal is to minimize the weight of the structure, determining the material type and orientation of each ply layer of every panel such that the loading constraints are satisfied locally and the panels form a

blended global design. Blending, in general, can be any measure of the manufacturability of a global design. Maintaining the continuity of material type and orientation across panel design boundaries, in part or in whole, is the primary concern in determining the blendedness of a global design. Ply orientation mismatches across panel design boundaries can cause manufacturing costs to rise in addition to the structural integrity issues associated with continuity breaks in the material.

The initial goal of this work is to introduce a scheme for blending the individual panel designs during the optimization process through modified migration similar to that introduced by McMahon [19]. Using the same approach implemented by Soremekun [30], it is assumed during the individual panel optimization that the loads on the individual panels are independent of the local panel designs. That is, load paths (the way loads are distributed across the panels of the structure) in the global structure do not change as the local panel stiffnesses vary. This assumption is, in many cases, hard to justify. However, it can be assumed that the changes in the load path may be computed after all the local panels are designed, and the process may be repeated iteratively until the change in local panel loads is negligible.

### 3.1 LAMINATE ENCODING

Each laminate must be encoded for use in the genetic algorithm. Following the coding scheme used by McMahon [19], integer values from zero to seven represent the orientation of each ply. The positive integers map to orientation angles 0, 15, 30, 45, 60, 75, and 90 degrees from one to seven, respectively, with the zero encoding representing an empty ply. Successive occurrences of a gene map to the orientation with alternating plus and minus signs, starting from the center of the (symmetric) laminate. For example, the first occurrence of a 2 encoding maps to  $15^\circ$ , the second 2 maps to  $-15^\circ$ , and so on. This is the meaning of balanced. Symmetric means that the stacking sequence is symmetric with respect to the center of the laminate, and therefore only half of the orientations actually need to be encoded. Similarly, the material type for each ply is represented by an integer, but here the material type is restricted to a single value meaning the material type for

each ply is the same. Since the number of plies used in the laminate varies from design to design, the genetic representation must have a way to reflect these differences. Using the zero encoding to represent an empty ply permits laminates with varying numbers of plies, but also admits invalid designs. Consider the invalid laminate represented by the encoding or chromosome

$$[1\ 2\ 3\ 6\ 2\ 1\ 0\ 2\ 4\ 2\ 5\ 7\ 3\ 0\ 3\ 1].$$

Zero interior genes, depicting empty plies within the chromosome, split the laminate into pieces that can only be interpreted physically by shifting the zeros to one side of the chromosome, as in

$$[1\ 2\ 3\ 6\ 2\ 1\ 2\ 4\ 2\ 5\ 7\ 3\ 3\ 1\ 0\ 0].$$

Thus at every iteration, designs that contain internal empty plies are modified to move zeros to the right producing physically meaningful encodings.

The standard genetic operators used in this work, *crossover* and *mutation*, are illustrated here to avoid confusion. The crossover operator takes two parent strings, divides them at a crossover point and recombines them into two children strings. For this example the crossover point is marked by the | symbol.

parent-1 chromosome	$[1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 1\ 1\ 1\ 1\ 1\ 0\ 0]$
parent-2 chromosome	$[2\ 2\ 2\ 2\ 2\ 2\ 2\ 2 2\ 2\ 2\ 2\ 0\ 0\ 0]$
child-1 chromosome	$[1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 2\ 2\ 2\ 2\ 0\ 0\ 0]$
child-2 chromosome	$[2\ 2\ 2\ 2\ 2\ 2\ 2\ 2 1\ 1\ 1\ 1\ 1\ 0\ 0]$

If the criteria for mutation is met, then a random position in the chromosome is selected and changed randomly. For this example the mutating gene is denoted by the hat symbol.

chromosome before mutation	$[1\ 2\ 3\ 6\ 2\ 1\ 2\ \hat{4}\ 2\ 5\ 7\ 3\ 3\ 1\ 0\ 0]$
chromosome after mutation	$[1\ 2\ 3\ 6\ 2\ 1\ 2\ \hat{3}\ 2\ 5\ 7\ 3\ 3\ 1\ 0\ 0]$

The nonstandard genetic operators used in this work (*addition*, *deletion*, *swap*, and *permutation*) are briefly illustrated here. The addition operator adds a gene at a random

position in the chromosome. For this example the gene 1 is inserted at the position marked by the | symbol.

chromosome before addition    [1 2 3 6 2 1 2 4|2 5 7 3 3 1 0 0]  
 chromosome after addition    [1 2 3 6 2 1 2 4 1 2 5 7 3 3 1 0]

The deletion operator deletes a gene from a random position within the chromosome. For this example the deleted gene is marked by the hat symbol.

chromosome before deletion    [1 2 3 6 2 1 2  $\hat{4}$  2 5 7 3 3 1 0 0]  
 chromosome after deletion    [1 2 3 6 2 1 2 2 5 7 3 3 1 0 0 0]

The swap operator switches the position of two randomly selected genes in the chromosome. Those genes participating in the swap are underscored.

chromosome before swap    [1 2 3 6 2 1 2 4 2 5 7 3 3 1 0 0]  
 chromosome after swap    [1 2 4 6 2 1 2 3 2 5 7 3 3 1 0 0]

The permutation operator randomly selects a section of the chromosome and inverts the order of all genes in that section. For this example the section to be permuted is enclosed with | symbols.

chromosome before permutation    [1 2 3 | 6 2 1 2 4 2 | 5 7 3 3 1 0 0]  
 chromosome after permutation    [1 2 3 | 2 4 2 1 2 6 | 5 7 3 3 1 0 0]

### 3.2 SELECTION STRATEGIES

An important aspect of the function of a GA appears in how individuals are selected to become parents of a child in the next generation. Since the goal of the algorithm is to simulate a survival of the fittest evolution strategy, we would like more fit individuals to have a higher probability of producing children. This is accomplished using a roulette wheel selection process common in the literature. The roulette wheel assigns to each individual of a population a probability of being selected according to its fitness ranking. The probability is biased such that the chance of selecting a particular individual is greater than choosing



another individual with lower fitness. The precise method for associating the probabilities is given by

$$\text{fraction of roulette wheel} = \frac{2(N + 1 - i)}{N^2 + N},$$

where  $i$  represents the  $i$ th best individual in a local population of interbreeding individuals or population of size  $N$ . The standard Fortran90 random number generator is used to provide approximate uniform distributions where random numbers are required to direct evolution.

A similar important function of the GA is in the evaluation of the parents and ways in which they are allowed to combine to create a child design. This is referred to as the crossover type. For all the runs presented in this work, a simple one-point-empty-thin crossover is implemented. A single crossover point falls within the parent string that has the largest number of empty genes or zero encodings. The two parent strings are then divided at that crossover point and the left piece from parent one is combined with the right piece from parent two. Similarly a second child is created with the right piece from parent one and the left piece from parent two. Although the crossover type can have a large impact on the variation of designs obtained from the initial population, the point here is simply to review the feasibility of the blending algorithm so no further crossover types are examined. However, to generate new designs throughout the generations, a small probability of mutation is added, allowing random genes in a child to change their orientation once crossover is complete. Child population members are required to be unique with respect to the generating parent population and members already added to the current child population. Once the child population is created, a selection scheme is invoked to choose the next generation from the union of the child population and the parent population. Results are presented for both multiple elitist selection and standard elitist selection in Chapter 4.

### 3.3 SERIAL GA PSEUDO CODE

For reference, a basic serial genetic algorithm is given here. Migration can be done serially by iterating through each independent population in turn, but it is more natural in the parallel context when subpopulations are evolving in parallel. The serial genetic algorithm, without migration, is:

```
initialize GA probabilities for crossover, mutation, addition, deletion, swap,  
    permutation  
gen := 0  
initialize Population(gen)  
decode and evaluate Population(gen)  
while termination not detected do  
    apply crossover to Population(gen) giving Children(gen)  
    apply genetic operators to Children(gen)  
    decode and evaluate Children(gen)  
    Population(gen + 1) := select from (Population(gen)  $\cup$  Children(gen))  
    gen := gen + 1  
    check for termination  
end do
```

### 3.4 EDIT DISTANCE

A metric is used to evaluate the similarities of evolution between adjacent panels. Immigrant individuals are compared, as strings of encoded stacking sequences, with each member of a population to determine the minimum distance between the individuals of the current generation and the migrant individuals (those received from neighboring populations). Those individuals reporting low correlation receive a penalty to their computed fitness value that decreases as correlation increases.

The distance metric used in this work is the Levenstein distance [16], more commonly known as the edit distance. It is applied in many situations where a quantitative measure of the similarity between strings is desired. The most common uses are in spelling correction algorithms and applications in computational biology comparing genomic sequence data [9].

The edit distance can be defined as the minimum number of edit operations (insertion, deletion, and substitution) required to transform one string into another. Consider the strings “SPOKEN” and “SPOTTED”. Let the characters I, D, S, M represent insertion, deletion, substitution, and matching character operations, respectively. The transformation

from “SPOKEN” to “SPOTTED” is not unique and could, for example, be done through the following operations:

M	M	M	D	I	I	M	D	I
S	P	O	K	-	-	E	N	-
S	P	O	-	T	T	E	-	D

The string MMMDIIMDI is called the edit transcript for the above transformation and contains five edit operations. The transcript is, however, not optimal in length. Alternatively, consider

M	M	M	S	I	M	S
S	P	O	K	-	E	N
S	P	O	T	T	E	D

where MMMSIMS is an optimal edit transcript containing only three edit operations.

The task of finding the minimum number of edit operations and the corresponding edit transcript is referred to as the edit distance problem, and is generally solved using dynamic programming [9]. For the current application, the edit distance value is used without the edit distance transcript to produce a blending measure suitable for modifying the fitness values of the composite laminate designs. The original fitness value  $F$  for each design is modified according to

$$\tilde{F} = \left( \left( \frac{d}{D} \right)^\alpha + 1 \right) F,$$

where  $d$  is defined to be the edit distance from a member to the set of immigrants from the neighboring panel.  $D$  is defined to be the number of active genes in the longer of the two chromosomes to be compared (maximum distance assuming all substitutions) and  $\alpha$ , the scaling factor, is a user-defined constant that dictates the influence of blending on the optimization run. If the scaling factor  $\alpha$  is large, then only blending will matter in the optimization; if the scaling factor is zero, then blending is ignored. In the present work the scaling factor is varied over a range to assess its influence on the resulting designs. The basic modification formula given above assumes the fitness value is some positive number with values closer to zero being more desirable.

Given any communication topology more complex than a ring in which each panel has only a single neighbor, the formula must be generalized to handle migrant information from multiple adjacent neighbors. In the previous formula the quotient  $d/D$  represented the extent to which a design correlated with the migrant information available or the correlation factor ( $CF$ ). In a multiple neighbor situation the  $CF$  is constructed piecewise from each communicating neighbor. For  $N$  communicating neighbors,

$$CF_1 = \left(\frac{1}{N}\right) \left(\frac{d_1}{D_1} + \frac{d_2}{D_2} + \dots + \frac{d_N}{D_N}\right),$$

where  $d_i/D_i$  is the blending quotient associated with neighbor  $i$ . So, the general form of the fitness modification becomes

$$\tilde{F}_1 = \left(\frac{\alpha}{|S|} \left(\sum_{i \in S} \frac{d_i}{D_i}\right) + 1\right) F,$$

with  $S$  being the set of all neighbors sending migrant information to the current node. An alternative to  $\tilde{F}_1$ , which captures the worst case ply orientation mismatch rather than the sum, is

$$\tilde{F}_\infty = \left(\frac{\alpha}{|S|} \left(\max_{i \in S} \frac{d_i}{D_i}\right) + 1\right) F.$$

The material and stacking sequence continuity measures used by Liu and Haftka [17] are designed to separate the part of the continuity that affects load redistribution from the part that does not. Ostensibly this allows a rational integration of the continuity measure into the global-local optimization, but ultimately the measure is still a heuristic. The edit distance is a true metric (in the precise mathematical sense), is very flexible via weighting the changes, and is well understood in many different scientific contexts.

An example is given here for changing the composite laminate stacking sequence encoding (4471620) into (4473361) for the case when zero encodings are considered to match any encoding (modified edit distance).

M	M	M	S	I	M	M
4	4	7	1	-	6	0
4	4	7	3	3	6	1

where MMMSIMM is an optimal edit transcript containing only two edit operations. The additional case for using the standard edit distance, in which the zero encoding is not considered a match for any other ply encoding, is precisely a string comparison using the encoded alphabet with no changes from the spoken to spotted example given previously.

### 3.5 PARALLEL GA PSEUDO CODE

The pseudo code in this section outlines the actions taken by each processing node in the parallel environment. For this work the parallel environment is primarily a cluster of machines (distributed memory) with a high speed interconnect but, since the code is written in MPI, can also be run on shared memory architectures. The point of parallelism in the algorithm is in some ways trivial as each processor controls a distinct population corresponding to a single panel. Without migration the algorithm would optimize each panel in isolation mimicking sequential serial runs. All of the interesting portions of the parallelization occur in the methods controlling migration. Since the panel optimizations are not required to proceed in lockstep, communication is handled asynchronously. It makes no difference if the panels are optimized on processors of different speeds, additional generations on faster processors only serve to increase the probability that better local designs are discovered early.

The whole point of migration in this algorithm is to provide each local population with some knowledge of evolutionary trends occurring in adjacent populations. It seems logical that more recent migrant information portrays neighboring evolution more accurately. The goal is the highest migration rate possible without swamping the system with communication, balancing analysis time with any delay caused by message passing. Compute bound asynchronous communication algorithms execute local analysis operations while information travels between processors, effectively reducing time wasted for communication to zero. In contrast, a communication bound asynchronous communication algorithm can not find enough work to do during message passing and is forced to idle until required messages have completed. The desired balance is to have as much migration as possible without forcing the algorithm to become communication bound. This balance is accomplished dynamically through the use of a circular queuing system characterized by sending and receiving modes. The circular queuing system is used to monitor and limit the number of messages allowed in the communication fabric at one time and is described in more detail below.

Each processor has the ability to send and/or receive migrant information from any number of user defined neighbors. The first step in reducing communication time when using

MPI, a commonly used message passing interface for parallel FORTRAN and C programs, is to ensure that receiving nodes post receive messages before their corresponding sends are posted to the system. The second step is to make sure that only a small number of incomplete message passing handshakes are in the system at any given time. The more incomplete messages allowed in the system the larger the lookup tables are for message completion and the longer each message passing handshake takes to complete individually. Migration queues provide the data to control the migration rate dynamically and provide the ground work for minimizing communication time of MPI spawned messages.

Each processor has a single send queue constructed as a circular queue containing at least `wait_iterations` nodes. At each iteration of the GA, with the prescribed probability of migration, an order for migration is placed in an empty node of the queue carrying the number of migrants to be sent and the iteration index the request was generated on. On the same iteration that the request is placed in the queue a short message is sent to every receiving neighbor of the current processor containing the number of migrants that are coming. In this way migration requests are added to the queue and receiving neighbors become aware of the size of incoming messages. The actual act of sending migrant individuals happens after a small number of iterations, `wait_iterations`, from the generation in which the request was made. This is a user defined constant and is set to give enough wall clock time for receivers to pre-post matching messages indicating the size and source of incoming migrant data, satisfying the first requirement for reduced communication time. This does not guarantee that the receives will be pre-posted before messages arrive, but simply raises the probability proportionally with the size of `wait_iterations`. To be clear, note that the migrants that are sent are the current top ranked individuals of the population, which are not necessarily the top ranked individuals appearing when the request was made. This portion of the algorithm is similar to predetermining iterations which contain a migration step without removing the random nature of when migration occurs and how many migrants are sent.

The receive portion of the queuing system is composed of an array of receive queues corresponding to each neighbor the current processor can receive migrants from. Since

the message from senders containing the number of migrants to be sent is a single integer and as such is constant in size, the receivers can pre-post sets of receive messages to wait for these warning messages. Each node of the receive queues is checked each round for incoming messages bringing information about the next set of migrants. Upon receiving this information, receive messages can be posted in anticipation of arriving migrants that identify the size, number of migrants, and sender of the data. The head of the queue is monitored for completed handshake operations and when messages are complete the receiver sends confirmation to the sender. This allows the send queue to free up a node and the receiver can continue checking the receive queues until an empty node is found or a message awaiting completion is found. Multiple migration completions can be handled at the receiving end in a single iteration, in which case only the most up to date migrants are stored for comparison.

The size of the send queue dictates how many uncompleted messages are allowed in the system at one time. Limiting the number of uncompleted messages floating around in the system bounds the time needed to complete a single communication. Migration requests that attempt to enter a full queue are dropped and recorded as failed migrations. A failed migration simply means that the communication time was exceeding the computation time, allowing the queue to fill to capacity with waiting migrations, and the algorithm was becoming communication bound. This method allows for the migration rate to be set arbitrarily high and be bound dynamically by the compute time used for analysis. Dropped migrations for all runs presented in this work are less than one percent of total migrations.

This method is a variation on the migration procedure of McMahon [20] in that migrant individuals are not allowed into the destination population by default. Migrants are stored in, for lack of a better term, a *migrant box* and are looked at when applying the blending metric. If an individual in the box can represent a valid member of the current population, then it is allowed to become a member, completing the full migration. If, however, a migrant is somehow invalid for the current population, then the full migration does not occur and the migrant remains in the box for comparison alone. The results presented in Chapter 4 disallow migrants to enter the population to allow evaluation of the blending algorithm alone. Pseudo code for the parallel GA with migration and blending follows.

```

initialize GA probabilities for crossover, mutation, addition, deletion, swap,
    permutation, and migration
initialize migration parameter wait_iterations
gen := 0
initialize Population(gen)
decode and evaluate Population(gen)
while termination not detected do
    begin migration
        if migration appropriate
            choose number of emigrants
            queue migration to occur at generation gen + wait_iterations
        end if
        if migration queued for this generation
            send emigrants (current top ranked individuals)
        end if
        if immigrants available
            receive immigrants in migrant_box
        end if
    end migration
    apply crossover to Population(gen) giving Children(gen)
    apply genetic operators to Children(gen)
    decode and evaluate Children(gen)
    apply blending metric to Children(gen) and Population(gen)
    if migrant_box designs are valid
        Population(gen + 1) := select from (migrant_box  $\cup$  Population(gen)  $\cup$  Children(gen))
    else
        Population(gen + 1) := select from (Population(gen)  $\cup$  Children(gen))
    end if
    gen := gen + 1
    check for termination
end do

```



## Chapter 4: REFERENCE MIGRATION RESULTS

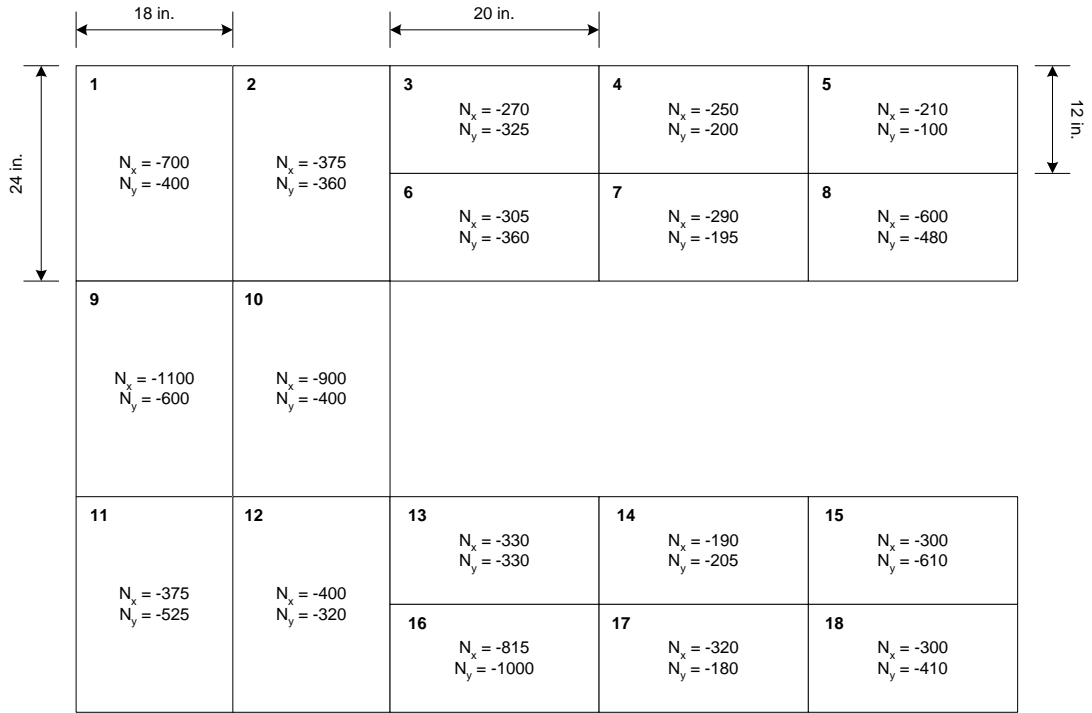


FIG. 2. Eighteen panel irregular grid test problem.

Results are given here for a test problem consisting of eighteen panels configured and loaded as shown in Figure 2 [30]. The panels can be composed of 0,  $\pm 15$ ,  $\pm 30$ ,  $\pm 45$ ,  $\pm 60$ ,  $\pm 75$ , and 90 degree ply orientations. One material type, graphite-epoxy (IM7/8552), is used for construction of each ply. The parameters given in Table 1 apply to each run presented in this chapter, are chosen based on the work of others [20], and are expected to be good choices for the genetic algorithm. They are, however, not critical to the performance of the algorithm.

The initial run was to fix a control set for the problem itself where each panel is optimized in isolation. The result is presented in Table 2 with the total weight of the structure being 27.92 kg. As a measure of manufacturability, the total edit distance between adjacent panels of the structure is tallied in two ways. The first is calculated according to strict edit distance while the second method uses a modified edit distance metric in which any comparison involving an empty ply (zero encoding, these plies are pushed to the surface

TABLE 1. *Static GA parameters.*

Probability of crossover	1.00
Probability of mutation (orientation)	0.05
Probability of ply addition	0.05
Probability of ply deletion	0.10
Probability of ply permutation	0.00
Probability of ply swap	0.00
Population size	100
Laminate ply genes	24
Number of elite retained	5
Termination criterion (generations)	2000
Probability of migration	0.50
Migration queue offset ( <i>wait_iterations</i> )	4

of the laminate) is considered a match. Measured in the first way the structure has a total edit distance of 242. The second method measures the edit distance at 90. Total edit distance is the sum of the edit distances between all neighboring panel pairs. Both methods consider panels to be neighbors if they are adjacent vertically or horizontally and indicate that the unblended design would be extremely difficult and expensive if not impossible to manufacture.

#### 4.1 RING COMMUNICATION TOPOLOGY

The initial blended design runs placed the 18 panels in a ring communication topology with data flowing in only one direction around the ring. Each panel communicated with the panel whose identification number (boldface numerals in Figure 2) directly followed it in increasing counting order while the 18th panel communicated with the first to complete the ring. Runs were executed with a uniform distribution of the scaling factor  $\alpha$  from 0.05 to 8 and had the maximum number of migrants in the migrant box set to 1. Recall that as the scaling factor  $\alpha$  increases, the impact of the blending measure increases by multiplying the fitness by a fraction of the largest possible distance between two neighbors and the scaling factor. This creates an increasingly severe penalty to panel fitness for panels that are not blended as alpha increases. At  $\alpha = 8$ , the structure converged to a perfectly blended design, using the standard edit distance metric, of

$$[30/ \pm 30/75/ \pm 75/ \pm 45_4/ \pm 60/ \pm 75/15]_s .$$

TABLE 2. Control run (no blending), edit distance 242, modified edit distance 90, total structural weight 27.92.

Panel	Weight (kg)	Design
1	2.68	$[\pm 30_8]_s$
2	2.35	$[\pm 30_7]_s$
3	0.93	$[\pm 75_2 / - 75 / - 60 / 75 / 60 / \pm 75]_s$
4	0.84	$[- 75 / 60 / 75 / - 60 / \pm 75 / - 75 / 60 / 75]_s$
5	0.74	$[60 / \pm 60 / 75 / - 75 / - 60 / 75 / 60]_s$
6	1.03	$[\pm 75_2 / - 75 / 60 / 75 / - 75 / \pm 60 / 75]_s$
7	0.84	$[60 / \pm 75 / \pm 60 / \pm 75_2]_s$
8	1.12	$[60 / \mp 75 / - 60 / 75 / \pm 75_2 / 60 / \pm 75]_s$
9	3.18	$[30 / \pm 30_9]_s$
10	2.85	$[- 45 / \pm 30_5 / - 30 / 15 / 30 / - 30 / 45 / 30]_s$
11	2.51	$[30 / \pm 30_7]_s$
12	2.35	$[\pm 30_7]_s$
13	1.03	$[\pm 75 / - 75 / 60 / 75 / - 75 / \pm 60 / 75 / \pm 75]_s$
14	0.84	$[75 / \pm 75 / - 60 / \pm 75 / 60 / \pm 75]_s$
15	1.11	$[\pm 75_6]_s$
16	1.67	$[\pm 75_9]_s$
17	0.84	$[60 / - 60 / - 75 / 60 / 75 / \pm 75_2]_s$
18	1.03	$[75 / \pm 75_5]_s$

Though the structure has a total edit distance of zero for both manufacturability measures (standard edit distance and modified edit distance) the weight of the structure increased to 40.32 kg. As a way to keep the weight of the structure down, the edit distance metric was modified in the code to consider any comparison with a zero encoding to be a match. This method also converged to a perfectly matched (zero modified edit distance) design in the ring topology, shown in Table 3. However, as would be expected, the perfect matching in the ring does not correspond to perfect matching in the actual problem topology. At  $\alpha = 4$  the design had converged in the ring topology to perfect matching with a weight of 29.21 kg. The modified total edit distance, considering empty ply encodings to match anything, for the structure in the true topology mapped adjacencies was 6. Note that a perfectly matched design, in the sense of having a total (modified) edit distance of zero, is not necessarily perfectly blended in the engineering sense of being able to manufacture the structure with the minimal number of layer deposition passes. The intent of the present

work is that small (mathematical) edit distances should lead to well blended designs in the manufacturing sense.

The above discussion, and the design reported in Table 3, are for a migrant box of size one. It might seem that a larger migrant box size could only help blending and convergence, by providing more opportunities for adjacent panel designs to blend. Paradoxically, this is *not* the case. The problem with multiple migrants is illustrated in Figure 3, which displays a simplified ring topology of the top five ranked individuals of a three panel example. The figure portrays a converged locally nonoptimal configuration in which the algorithm reports perfect matching for the ring. That is to say, the edit distance  $d$  from the top ranked individual of each panel to the migrant set from its corresponding neighbor is found to be zero. In the example shown however, the panels' top designs are matching perfectly with lower ranked designs in the migrant boxes and the blended fitness clearly does not impose blending constraints across the rank one individuals, which are taken as the final design. Restricting the migrant box to size one eliminates this flaw, but sending only single migrants then enormously restricts the flow of information between panels.

#### 4.2 PHYSICALLY MATCHED COMMUNICATION TOPOLOGY

To capture the true adjacency configuration the ring topology was discarded and a full communication structure was implemented mirroring the test problem topology. Every panel communicated data to each panel adjacent to itself. After hundreds of test runs (each for thousands of generations) using  $\alpha$  values varying from 0.05 to 1000 (and migrant box size equal to one) the method did not converge to a matched design, but rather stagnated in false local optima created by the communication topology itself.

The A-B diagram inset of Figure 4 portrays the underlying problem with the natural expansion of the method to topology mapped communication. The A and B of the diagram represent two completely unmatched designs with respect to each other. However, the configuration of this example is a converged design, since any panel with the A design has at least two neighbors also having the same design and at most one neighbor having the B design. This presents a situation in which, though there is no blending between panels

TABLE 3. Ring topology blending, modified (ring) edit distance 0, modified (true topology) edit distance 6, maximum modified edit distance between adjacent panels 4, total structural weight 29.21.

Panel	Weight (kg)	Design						
1	2.85	[ 30	±30	±30	-30	75	±75	-75/ ± 75/ ± 30/90/75/30] <sub>s</sub>
2	2.45	[	30	±30	-30	75	±75	-75/ ± 75/ ± 30/90/75/30] <sub>s</sub>
3	1.03	[				75	±75	-75/ ± 75/ ± 30/90/75/30] <sub>s</sub>
4	0.84	[				75		-75/ ± 75/ ± 30/90/75/30] <sub>s</sub>
5	0.74	[						-75/ ± 75/ ± 30/90/75/30] <sub>s</sub>
6	1.03	[				75	±75	-75/ ± 75/ ± 30/90/75/30] <sub>s</sub>
7	0.93	[					±75	-75/ ± 75/ ± 30/90/75/30] <sub>s</sub>
8	1.12	[			60	75	±75	-75/ ± 75/ ± 30/90/75/30] <sub>s</sub>
9	3.35	[ ±30 <sub>2</sub>	±30	±30	60	75	±75	-75/ ± 75/ ± 30/90/75/30] <sub>s</sub>
10	3.02	[ ±30	±30	±30	60	75	±75	-75/ ± 75/ ± 30/90/75/30] <sub>s</sub>
11	2.68	[	±30	±30	60	75	±75	-75/ ± 75/ ± 30/90/75/30] <sub>s</sub>
12	2.51	[	-30	±30	60	75	±75	-75/ ± 75/ ± 30/90/75/30] <sub>s</sub>
13	1.03	[				75	±75	-75/ ± 75/ ± 30/90/75/30] <sub>s</sub>
14	0.84	[				75		-75/ ± 75/ ± 30/90/75/30] <sub>s</sub>
15	1.11	[			-75	75	±75	-75/ ± 75/ ± 30/90/75/30] <sub>s</sub>
16	1.67	[ ±75	±75	±75	-75	75	±75	-75/ ± 75/ ± 30/90/75/30] <sub>s</sub>
17	0.93	[					±75	-75/ ± 75/ ± 30/90/75/30] <sub>s</sub>
18	1.03	[				75	±75	-75/ ± 75/ ± 30/90/75/30] <sub>s</sub>

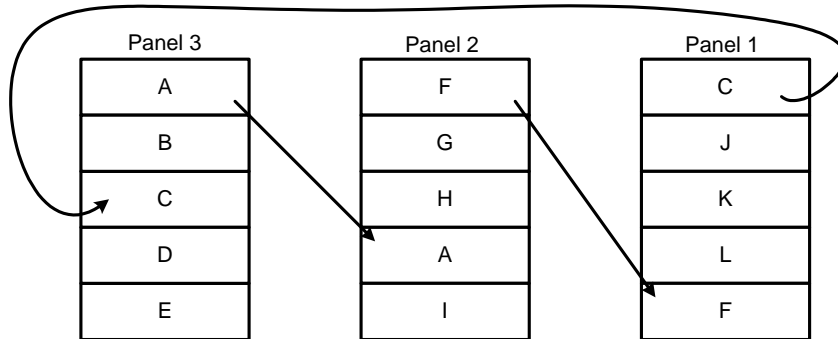


FIG. 3. Migrant box of size five, for ring 1 → 2 → 3 → 1. The top five designs of each panel become its emigrants.

of design A and B, none of the panels can find a more blended design than they already

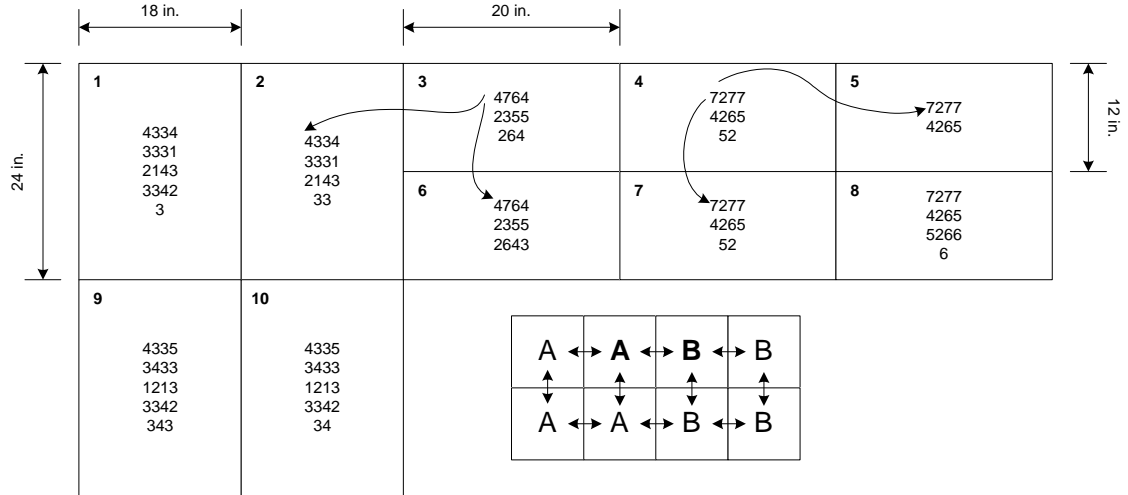


FIG. 4. *False optimum of topology mapped communication. Each panel shows the final (best) converged design, written in groups of four encodings read left to right top to bottom.*

have. If a panel with configuration A changes anything to become more like B then it gains favor with one neighbor and loses favor with two neighbors. A more complex illustration of this phenomenon is displayed in Figure 4 proper, which illustrates a converged globally nonoptimal configuration obtained using topology mapped communication paths. Each panel sends and receives migrant data (migrant box size is one here) from all panels directly adjacent to itself. The design displayed is a converged design with  $\alpha = 1000$ . By making  $\alpha$  large, the algorithm is forced to try and perfectly match the panels as any global design with a mismatched ply orientation between neighbors would force the modified fitness of the involved panels to become quite unfit.

Consider Panel 3 and Panel 4 of Figure 4. The first (innermost) ply angle of Panel 3 is encoded with a value of four. Clearly Panel 4 is adjacent to Panel 3, so why do their ply orientations not match even with a scaling factor of 1000? The answer lies in how the modification to the fitness function is calculated for multiple neighbors. Each neighbor of Panel 3 has an equal influence on the calculated fitness of the individuals in the population as described by the equations in Chapter 3.4. Panel 4 has two of three neighbors that are perfectly matched with it in accordance with the modified edit distance metric allowing empty plies to match anything. This puts Panel 4 in a position similar to the A-B diagram

insert in which it is unwilling to modify its design, since it is already fully supported perfectly by two of three neighbors. Panel 3 is also unwilling to change to be more blended with Panel 4 as it can find no design that lowers the total edit distance from itself to its neighbors. This converged locally nonoptimal configuration is formed strictly from the fact that each panel is trying to conform with up to four different neighbors simultaneously, and the entire process does not in general converge to a matched design, or even a locally optimal design. Recall that the migrants were used to modify the fitness value of a population’s individuals, but the migrants did not actually enter the population. Letting the migrants enter the populations makes no substantive difference—the GA still fails to converge to matched designs for the same reasons described above. Similarly, using the maximum mismatch fitness  $\tilde{F}_\infty$  rather than the mismatch sum fitness  $\tilde{F}_1$  makes no qualitative difference. Figure 6 later shows that  $\tilde{F}_1$  and  $\tilde{F}_\infty$  are equally poor.

This nonconvergence is not surprising, since what is being attempted here is similar to concurrent subspace optimization (CSSO), a well known technique whereby a system optimization is attempted by coordinating independent subsystem optimizations. The fitness modification  $\tilde{F}$  to reflect matching is tantamount to the global coordination step in CSSO. CSSO, and numerous variants of it, are known to fail to converge, both in theory and practice [28]. Nevertheless, paradigms do exist wherein independent subsystem optimizations can be guided to an overall optimal system design (see, e.g., [25], [26]). Using these other paradigms will require a major reformulation of the present blending problem—this remains a topic for future research.

### 4.3 WEIGHT-COST TRADEOFFS

Even though the genetic algorithm may not converge to a completely blended overall structure, it does nevertheless produce many good (light) designs for the panels with varying degrees of blending. It is useful to the design engineer to have this information in the form of trade-off plots, as in Figure 5, where a higher edit distance on the  $x$  axis roughly indicates an increasing cost to manufacture in exchange for a lower weight. The scatter plots (Figure 5 a.–d.) show the cost-weight trade-off for varying scaling factors for the four types of

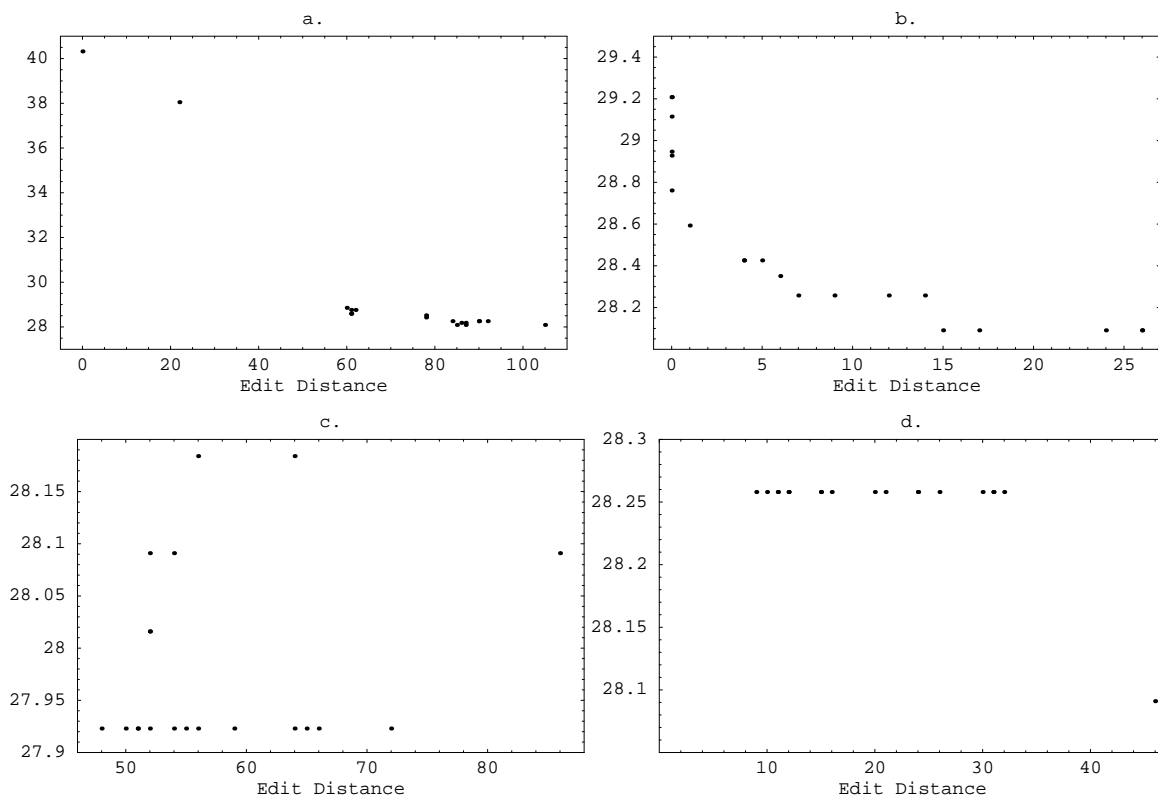


FIG. 5. *The scatter plots (a.–d.) show the cost-weight trade-off for varying scaling factors for the four types of runs described above. In each graph the y-axis is weight and the x-axis is the sum of the edit distances between all communicating pairs. a. Unmodified edit distance in a ring topology passing single migrants. b. Modified edit distance, in which empty ply encodings are considered to match, in a ring topology passing single migrants. c. Modified edit distance and topology mapped communication grid, where each panel sends and receives data from each adjacent panel, passing single migrants. d. Modified edit distance using multiple migrants in the migrant box, ring topology.*

runs described above. In each graph the  $y$ -axis is weight and the  $x$ -axis is the sum of the edit distances between all communicating pairs. Figures 5a and 5b show the ring topology results for the two ways of measuring edit distance, and both show a design with zero (ring) edit distance as measured by their two respective edit distance metrics (perfectly matched with respect to the ring topology). The designs in Figure 5b illustrate the tradeoff between weight and matching—the lowest weight (28.59) of these that has a true topology modified edit distance (total number of ply orientation mismatches) less than 10 is given in Table 4. A heavier (29.21) design having a true topology modified edit distance of 5 is given in Table 5. It is interesting to note that the design in Table 4 is manufacturable even though it is not



TABLE 4. *Ring topology blending, modified (ring) edit distance 1, modified (true topology) edit distance 8, maximum modified edit distance between adjacent panels 5, total structural weight 28.59.*

Panel	Weight (kg)	Design						
1	2.85	[	$\pm 30_2$	$\pm 30$	-75	75	-75	$\pm 60_3/75/15]_s$
2	2.51	[	$\pm 30$	$\pm 30$	-75	75	-75	$\pm 60_3/75/15]_s$
3	0.93	[				75	-75	$\pm 60_3/75/15]_s$
4	0.84	[					-75	$\pm 60_3/75/15]_s$
5	0.74	[						$\pm 60_3/75/15]_s$
6	1.02	[			-75	75	-75	$\pm 60_3/75/15]_s$
7	0.84	[					-75	$\pm 60_3/75/15]_s$
8	1.12	[			-75	75	60	-75 $\pm 60_3/75/15]_s$
9	3.18	[	$\pm 30$	$\pm 30_2$	$\pm 30$	75	60	-75 $\pm 60_3/75/15]_s$
10	3.02	[	30	$\pm 30_2$	$\pm 30$	75	60	-75 $\pm 60_3/75/15]_s$
11	2.51	[		$\pm 30$	$\pm 30$	75	60	-75 $\pm 60_3/75/15]_s$
12	2.51	[			$\pm 30$	75	60	-75 $\pm 60_3/75/15]_s$
13	1.02	[				75	60	-75 $\pm 60_3/75/15]_s$
14	0.84	[					-75	$\pm 60_3/75/15]_s$
15	1.12	[			75	-75	75	-75 $\pm 60_3/75/15]_s$
16	1.68	[	75	$\pm 75_2$	$\pm 75$	-75	75	-75 $\pm 60_3/75/15]_s$
17	0.84	[					-75	$\pm 60_3/75/15]_s$
18	1.02	[				-75	75	-75 $\pm 60_3/75/15]_s$

perfectly blended, and it is lighter than the best (29.21 kg) fully blended design found by Sormekun et al. [30]. It is manufacturable because the number of ply mismatches between panels is relatively low. However, there is no recognized definition of manufacturability.

Figure 5d shows the failure of multiple migrants illustrated in Figure 3. Note the large edit distances at the same weight; these are converged nonoptimal designs. Figure 5c shows the results for the communication topology matching the problem topology. None of these designs are easily manufactured because of the large numbers of ply mismatches (edit distances).

Just as incorporating migrants into the population rather than holding them in a migrant box for comparison makes no significant difference in performance, the two mismatch metrics  $\tilde{F}$  (sum of edit distances) and  $\tilde{F}_\infty$  (maximum edit distance) perform essentially the same. This is clear from Figure 6, which shows results for both  $\tilde{F}_1$  (plain font) and  $\tilde{F}_\infty$  (bold

TABLE 5. *Ring topology blending, modified (ring) edit distance 0, modified (true topology) edit distance 5, maximum modified edit distance between adjacent panels 3, total structural weight 29.21.*

Panel	Weight (kg)	Design						
1	2.85	[ 30	±30	±30	-30	75	±75	∓45/ - 75/ ∓ 60/60/45/75] <sub>s</sub>
2	2.51	[	30	±30	-30	75	±75	∓45/ - 75/ ∓ 60/60/45/75] <sub>s</sub>
3	1.02	[				75	±75	∓45/ - 75/ ∓ 60/60/45/75] <sub>s</sub>
4	0.84	[					75	∓45/ - 75/ ∓ 60/60/45/75] <sub>s</sub>
5	0.74	[						∓45/ - 75/ ∓ 60/60/45/75] <sub>s</sub>
6	1.02	[				75	±75	∓45/ - 75/ ∓ 60/60/45/75] <sub>s</sub>
7	0.93	[					±75	∓45/ - 75/ ∓ 60/60/45/75] <sub>s</sub>
8	1.12	[			-75	75	±75	∓45/ - 75/ ∓ 60/60/45/75] <sub>s</sub>
9	3.35	[ ±30 <sub>2</sub>	±30	±30	-75	75	±75	∓45/ - 75/ ∓ 60/60/45/75] <sub>s</sub>
10	3.02	[ ±30	±30	±30	-75	75	±75	∓45/ - 75/ ∓ 60/60/45/75] <sub>s</sub>
11	2.68	[	±30	±30	-75	75	±75	∓45/ - 75/ ∓ 60/60/45/75] <sub>s</sub>
12	2.51	[	30	±30	-75	75	±75	∓45/ - 75/ ∓ 60/60/45/75] <sub>s</sub>
13	1.02	[				75	±75	∓45/ - 75/ ∓ 60/60/45/75] <sub>s</sub>
14	0.84	[					75	∓45/ - 75/ ∓ 60/60/45/75] <sub>s</sub>
15	1.12	[			-75	75	±75	∓45/ - 75/ ∓ 60/60/45/75] <sub>s</sub>
16	1.68	[ ±75	±75	±75	-75	75	±75	∓45/ - 75/ ∓ 60/60/45/75] <sub>s</sub>
17	0.93	[					±75	∓45/ - 75/ ∓ 60/60/45/75] <sub>s</sub>
18	1.02	[				75	±75	∓45/ - 75/ ∓ 60/60/45/75] <sub>s</sub>

font). The (true topology edit distance, weight) data points are plotted as digits showing the maximum distance between any two neighbors. Even though a maximum distance of three is acceptable, the total edit distance (45 mismatches) for that design is not.

#### 4.4 UMBRA MIGRATION

Analysis of results reported in Chapter 4.1 showed that the unmodified edit distance, when used in a ring communication topology, could converge to zero (corresponding to ideal manufacture) with  $\alpha = 8$ . These results were both promising and foreboding. They are promising in that complete convergence to zero edit distance in such a large design space proves that the pressure to evolve towards your neighbors can be a powerful guide for evolution. The weights of designs developed with this method, however, were prohibitive, forcing the introduction of the modified edit distance. Though the modified edit distance

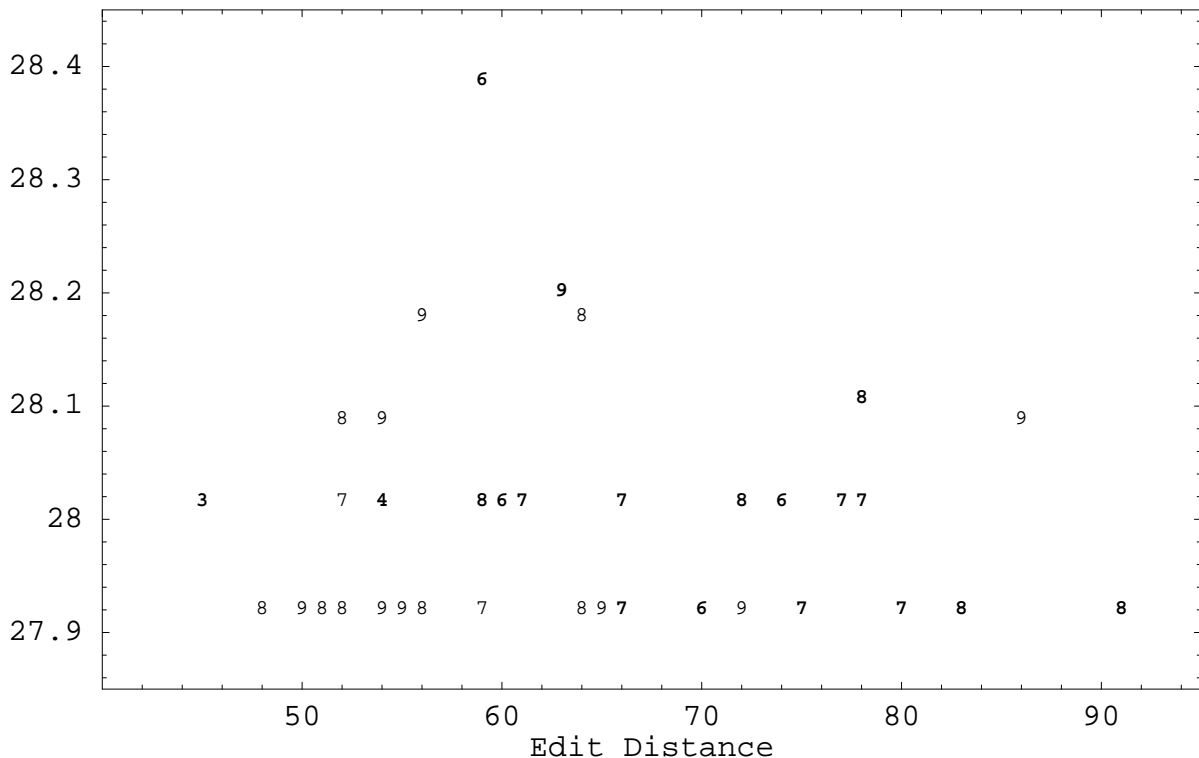


FIG. 6. The  $y$ -axis is weight. Each plotted point is rendered as an integer representing the maximum edit distance between any single pair of neighbors. Those points in non-bold font are the same points appearing in Figure 5c. They represent the topology mapped communication grid using modified edit distance and single migrant passing. Points in bold face represent runs with the same parameters but use  $\tilde{F}_\infty$  rather than  $\tilde{F}_1$  to modify the objective function value as described in Chapter 3.4.

allowed the designs to become much lighter, the ability for large  $\alpha$  to force blending across the true topology of the problem was lost. This prompted the extension to physically matched topology communication, which failed to converge for reasons given in Chapter 4.2.

Umbra migration is a combination of the ring design methods, introduced to capture the complete physically matched topology convergence of the unmodified edit distance, while maintaining the low weight designs discovered using the modified edit distance. The initial ring blending solutions, though prohibitive in weight, had the property that zero distance around the ring was equivalent to perfect blending across the physically matched topology. That is, two designs could have zero edit distance if and only if they were precisely equal at every ply layer including empty plies. Modified edit distance allowed empty (coded as

zero) plies to match any other ply orientation, creating the possibility for information to be lost in a ring traversal. If at any point in the ring a thinner panel follows a thick panel, the thinner panel can perfectly blend with the thicker panel using empty plies. Further, if a thicker panel follows the thinner panel, then the thicker one can generate new (arbitrary) ply orientations for all positions matching an empty ply. This amounts to a loss of information as migrants are passed around the ring and prevents the forced convergence to zero edit distance in the physically matched topology.

Umbra migration is defined by the modification of migrating individuals to include the ply orientations of the design found in the migrant box replacing all empty ply layers of the top ranked individual with any layers that may be thicker than the current design. If a migrating individual contains 27 non-empty ply layers and the migrant box design contains 30 non-empty ply layers then the migrating individual would take the first 27 layers of the top individual and the three remaining layers from the design in the migrant box to form the design that is actually transmitted as migration. This causes a shadow (*umbra*) “best” design for the outer plies of a laminate to completely traverse the ring ensuring that no information is lost through thick-thin-thick movements. In addition, umbra migration allows for completely converged designs that have zero ring edit distance to correspondingly have zero physically matched topology edit distance. One such design is given in Table 6. The design shown used  $\alpha = 1.7$ . Notice that every panel design in Table 6 is a subsequence of the largest panel in the structure, Panel 9, indicating that umbra migration was used. The fact that umbra migration provided the best results gave birth to a new idea that could take advantage of having all panels built as a subsequence of the largest design. This idea, guide based design, is detailed in the next chapter.

TABLE 6. *Ring topology blending with umbra migration, modified (ring) edit distance 0, modified (true topology) edit distance 0, total structural weight 29.19 kg.*

Panel	Weight (kg)	Design									
1	2.85	[			$\pm 30$	45	$\pm 45$	75	$\pm 75$	-60	$60 / \pm 60 / 90_4 / 15]_s$
2	2.51	[				45	$\pm 45$	75	$\pm 75$	-60	$60 / \pm 60 / 90_4 / 15]_s$
3	0.93	[							75	-60	$60 / \pm 60 / 90_4 / 15]_s$
4	0.84	[								-60	$60 / \pm 60 / 90_4 / 15]_s$
5	0.74	[									$60 / \pm 60 / 90_4 / 15]_s$
6	1.02	[							$\pm 75$	-60	$60 / \pm 60 / 90_4 / 15]_s$
7	0.84	[								-60	$60 / \pm 60 / 90_4 / 15]_s$
8	1.12	[						75	$\pm 75$	-60	$60 / \pm 60 / 90_4 / 15]_s$
9	3.35	[	$\pm 30$	75	$\pm 30$	45	$\pm 45$	75	$\pm 75$	-60	$60 / \pm 60 / 90_4 / 15]_s$
10	3.18	[	30	75	$\pm 30$	45	$\pm 45$	75	$\pm 75$	-60	$60 / \pm 60 / 90_4 / 15]_s$
11	2.68	[			30	45	$\pm 45$	75	$\pm 75$	-60	$60 / \pm 60 / 90_4 / 15]_s$
12	2.51	[				45	$\pm 45$	75	$\pm 75$	-60	$60 / \pm 60 / 90_4 / 15]_s$
13	1.02	[							$\pm 75$	-60	$60 / \pm 60 / 90_4 / 15]_s$
14	0.84	[								-60	$60 / \pm 60 / 90_4 / 15]_s$
15	1.12	[						75	$\pm 75$	-60	$60 / \pm 60 / 90_4 / 15]_s$
16	1.77	[	30	75	$\pm 30$	45	$\pm 45$	75	$\pm 75$	-60	$60 / \pm 60 / 90_4 / 15]_s$
17	0.84	[								-60	$60 / \pm 60 / 90_4 / 15]_s$
18	1.02	[							$\pm 75$	-60	$60 / \pm 60 / 90_4 / 15]_s$

## Chapter 5: GUIDE BASED DESIGN

The edit distance blending method defined by Adams et al. [2] is briefly described here to highlight the identifying characteristics of the algorithm and to provide a stage to present guide based design. The method is built upon the idea of a measure of blendedness. A metric, the edit distance, is used to determine the degree of blendedness when two panel stacking sequences are compared with each other. This measure can then be used to alter the fitness function of a GA, rewarding designs that have a higher degree of blendedness with a reference design.

The algorithm utilizes a GA optimization on each panel using predetermined local loading constraints. Each panel optimization is allocated to a single processing node in a parallel processing environment and is run asynchronously with respect to other panels. Migrant individuals are sent out at random to neighboring nodes and stored for reference. The migration is referred to as reference migration since the migrants do not enter the target population. Instead, migrants are used to modify the fitness function of the target panel optimization through the measure of blendedness. Those designs that are evolving closely with neighboring panel designs are rewarded with better fitness values and are more likely to create children. This creates the possibility for a localized optimization to converge to a globally blended design. The method met with limited success. Though it was used to discover the best known global design for the 18 panel problem presented in Figure 2, convergence to local optima prevented the algorithm from increasing the quality of the solution with added iterations/generations. Additionally, designs that were not perfectly blended were in general locally optimal designs that had no practical utility.

Some of the major features of this edit distance based blending approach can be summarized as:

- asynchronous, real time algorithm,
- multiple populations each running an independent genetic algorithm,
- measure of blendedness (edit distance) allows for nearly perfectly blended designs to result from a run that are manufacturable,

- blending accomplished through evolutionary pressures of reference migration so that unblended designs evolve to global blending,
- convergence is dictated by the scaling factor [2] and does not benefit from increased iterations of the algorithm.

A simplified definition of blending is used in this work. Two adjacent panel designs are blended if one design is obtained by deleting a contiguous series of outermost plies from the other. A global design is perfectly blended if and only if blending holds for every pair of adjacent panels. This simplified definition serves to reduce the search space of the problem and provide equal footing for comparisons with previous work on the same test case [2][30]. The creation of a guide based GA design process grew out of the need to limit the optimization to consider only those designs that are globally blended.

Using the simplified definition of blending, a perfectly blended global design can be generated by designing each local panel as an initial sequence ( $a_1, \dots, a_k$  is an *initial sequence* of the sequence  $a_1, \dots, a_k, \dots, a_n$ ) of a single larger guide design. A GA can be created to operate on and generate a population of individuals that guide the global design process so that only globally blended designs are in the problem domain. During the analysis phase of the GA a guide design (a single individual from the population) is evaluated to determine the optimal initial sequence for each local panel satisfying loading constraints and minimizing weight. This is accomplished by stripping ply layers from the guide design, starting from the outermost layers, one layer at a time and analyzing the resulting designs according to the constraints for each local panel. (More efficient variations of this are possible, e.g., starting with the initial sequence length equal to that of the last optimal design analyzed for this panel, and then adding/subtracting layers from this length.) The guide designs in the population acquire fitness values through the combination of optimal local designs generated by the guide itself so that each guide design has a single best series of initial sequences corresponding to each panel that can be used to construct a usable fitness function value. An optimal initial sequence is discovered for each local panel and the fitness of the global design becomes the sum of the individual fitness values for every optimized local panel. This modified analysis method provides the means to assign fitness

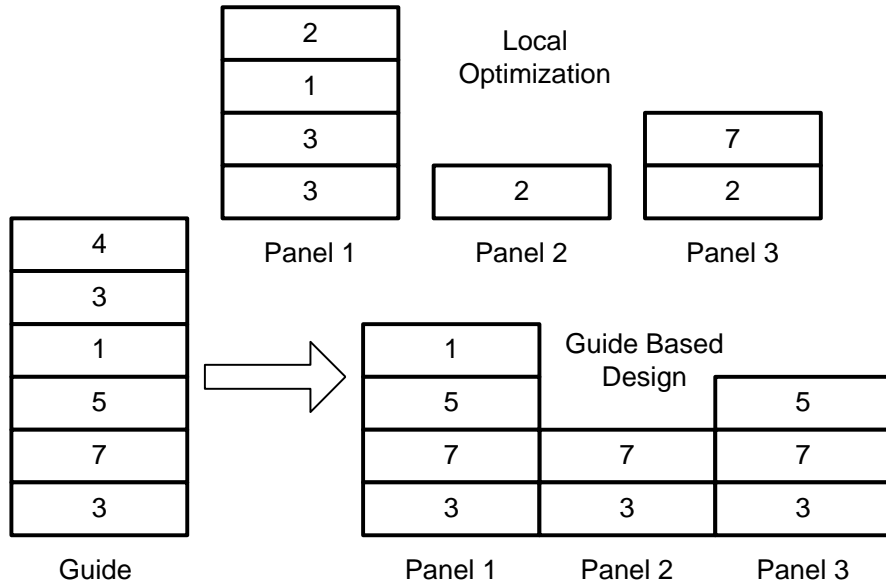


FIG. 7. *Guide based optimization example. The figure illustrates how isolated optimization creates designs that are locally optimal but are not easily manufacturable due to the ply orientation mismatches at through-the-thickness layers of the laminate. A side view of three sample panels, listing ply orientation encodings for each layer, is presented for both local optimization and a possible guide based design. Isolated local optimization generally produces light weight designs as less ply layers are required for Panels 2 and 3. The guide based design is optimized and constructed from the 6-ply thick guide design given on the left. The optimal initial sequence for the given guide produces a global design that is completely blended and therefore manufacturable. By operating on populations of guide designs a GA can discover an optimal blended global design minimizing weight and maximizing constraint satisfaction margins.*

values to guide designs themselves and allows a GA to operate on populations of guide designs to drive a global optimization process to an optimal blended design. An illustrative fictitious example of guide based design is presented in Figure 7.

As each guide design is assigned a fitness value, an accompanying array of integers is generated to indicate how many ply layers are applied to each panel. This array is used to reconstruct the optimal design discovered for an analyzed guide for each panel and becomes a table of information associated with each generation of the population.

A simple master-slave parallel code is implemented for this work. The master process generates and runs the GA code while distributing guide designs to slave processes for analysis in a lock step message-passing phase. The load distribution is calculated statically as an equal division of slave processes into population size with remaining work given



arbitrarily to selected nodes with lower rank. For a population size of 100, 3 slave processes were used distributing 34 guides to slave process 1 and 33 guides to slave processes 2 and 3.

Some of the major features of this new guide design based approach can be summarized as:

- synchronous, lock step algorithm,
- single population running a distributed genetic algorithm,
- nearly blended designs are not possible,
- blending accomplished directly through generation from a guide design to create globally blended designs,
- convergence is dictated by the GA parameters and allows additional generations to increase the probability that the top ranked solution is indeed optimal.

Observe that these characteristics are all notably different from the characteristics itemized for the edit distance approach.

## 5.1 MASTER-SLAVE PARALLELISM

Master-slave implementations are the simplest and most common form of parallel genetic algorithms where a single processor controls the genetic algorithm and the determination of fitness for each member of the population occurs in parallel by worker nodes. The classification of master-slave encompasses a wide variety of design decisions that affect the performance and scalability of the algorithm. A nice overview of parameter evaluation for parallel genetic algorithms can be found in the collection of work by Erick Cantú-Paz [7] in addition to the classical parallel algorithm design overview found in Quinn [23]. For the application developer, work load and communication time balancing are the primary concerns for applying a master-slave paradigm. General practice assigns the master process all the duties associated with the generic GA while distributing analysis work (fitness evaluation) to slave nodes.

Though a static distribution of work can perform well using deterministic non-iterative analysis routines, a more robust and scalable distribution method is required for complex analyses. Since the time required for a single analysis can vary greatly in iterative methods,

a dynamic producer-consumer model is more appropriate. That is, the master generates a queue of available work at each generation that is then distributed in small subsets to those slave processes requesting work. The slave processes consume work units and request more when they have completed their subset. For the current problem the smallest subdivision of labor, without parallelizing the analysis itself, is a single analysis of a local panel design. The work queue then at each generation becomes population size multiplied by the number of panels and the number of plies. For the 18 panel design problem presented in Figure 2 there are 43,200 ( $100 \times 18 \times 24$ , see Table 1) work units per generation.

The determination of how many jobs are distributed per request to those processes requesting work is part of the balance of communication and computation time. Similarly the boundaries on which work is distributed can relieve the master process from a large portion of data gathering. Dividing the work of a guide design analysis along panel division boundaries allows slave processes to only report the optimal local design for a given panel. Otherwise, the master process would be required to compute a min/max for each panel and choose the one most fit. Requiring this work from the master process increases the communication load by a factor of  $N$ , where  $N$  is the number of ply layers in a guide. The work load chunk then becomes the number  $W$  of panel analyses distributed to each processor such that the maximum tolerance for wasted compute time is bounded by the time  $T$  it takes to evaluate  $W$  designs. The worst case scenario is that all slave processes request work simultaneously for a remaining single unit of size  $W$ . In this case, all but one process remains idle for  $T$  units of time.

Guide based design is an easily parallelized and powerful method for dealing with large problems when the loads on each panel can be assumed fixed. The fact that local loads are assumed fixed for each panel can be viewed as a weakness of this algorithm but the speed and accuracy with which it can derive a solution to the problem allow for a possibly iterative process in which new loads can be introduced to refine the search and find better solutions.

## Chapter 6: GUIDE BASED DESIGN RESULTS

Results are given here for a test problem consisting of eighteen panels configured and loaded as shown in Figure 2 [30]. The panels are composed of 0,  $\pm 15$ ,  $\pm 30$ ,  $\pm 45$ ,  $\pm 60$ ,  $\pm 75$ , and 90 degree ply orientations. One material type, graphite-epoxy (IM7/8552), is used for construction of each ply. Table 7 summarizes the parameters of the simple GA operating on guide designs.

TABLE 7. *Static GA parameters.*

Probability of crossover	1.00
Probability of mutation (orientation)	0.05
Population size	100
Laminate ply genes	24
Number of elite retained	1
Termination criterion (generations)	2000

For comparison, each of the 18 panels is isolated and optimized locally without respect to a blending measure. The addition of blending constraints on the problem restricts possible global designs and allows individual panel optimization to serve as a lower bound for global weight. Though the global design obtained from isolated local panel optimization is not blended, the reference point given by the weight is useful in determining the quality of a blended global design. The resulting global design is presented in Table 2 with a total structural weight of 27.92 kg [2].

Sormekun et al. [30] discovered a blended design, using a less restrictive blending measure, with a total structural weight of 29.21 kg. The method, as briefly described earlier, used an iterative process to fix layers of the global design based on local designs with the fewest ply layers. Though the method was less restrictive with respect to blending than the methods in [2] and here, it serves as a reference to determine the quality of other blended designs.

A slightly better design was discovered using the edit distance blending metric with umbra migration in previous work by Adams et al. [2]. The edit distance approach creates multiple populations, each running an independent local optimization for an individual panel, using migration to transfer evolutionary data between neighboring panels. That

TABLE 8. Guide based design with total structural weight 28.63 kg.

Panel	Ply Encodings	Design								
1	17	[	$\pm 45$	30	$\mp 30$	45	$\mp 45$	60	$-60 / \pm 60_2 / -75 / 60 / 75]_s$	
2	14	[			$\mp 30$	45	$\mp 45$	60	$-60 / \pm 60_2 / -75 / 60 / 75]_s$	
3	11	[					$\mp 45$	60	$-60 / \pm 60_2 / -75 / 60 / 75]_s$	
4	9	[						60	$-60 / \pm 60_2 / -75 / 60 / 75]_s$	
5	8	[							$-60 / \pm 60_2 / -75 / 60 / 75]_s$	
6	11	[					$\mp 45$	60	$-60 / \pm 60_2 / -75 / 60 / 75]_s$	
7	9	[						60	$-60 / \pm 60_2 / -75 / 60 / 75]_s$	
8	13	[			30	45	$\mp 45$	60	$-60 / \pm 60_2 / -75 / 60 / 75]_s$	
9	19	[	$\pm 45$	$\pm 45$	30	$\mp 30$	45	$\mp 45$	60	$-60 / \pm 60_2 / -75 / 60 / 75]_s$
10	18	[	$-45$	$\pm 45$	30	$\mp 30$	45	$\mp 45$	60	$-60 / \pm 60_2 / -75 / 60 / 75]_s$
11	15	[			30	$\mp 30$	45	$\mp 45$	60	$-60 / \pm 60_2 / -75 / 60 / 75]_s$
12	14	[				$\mp 30$	45	$\mp 45$	60	$-60 / \pm 60_2 / -75 / 60 / 75]_s$
13	11	[					$\mp 45$	60	$-60 / \pm 60_2 / -75 / 60 / 75]_s$	
14	9	[						60	$-60 / \pm 60_2 / -75 / 60 / 75]_s$	
15	13	[				30	45	$\mp 45$	60	$-60 / \pm 60_2 / -75 / 60 / 75]_s$
16	19	[	$\pm 45$	$\pm 45$	30	$\mp 30$	45	$\mp 45$	60	$-60 / \pm 60_2 / -75 / 60 / 75]_s$
17	9	[						60	$-60 / \pm 60_2 / -75 / 60 / 75]_s$	
18	11	[					$\mp 45$	60	$-60 / \pm 60_2 / -75 / 60 / 75]_s$	

is, each panel is optimized locally in real time by an independent genetic algorithm while modifying the objective function values of individuals based on a blending metric comparison with migrants from adjacent evolving populations. A modified edit distance metric serves to measure the blendedness of adjacent panels by counting the number of edit operations required to turn the encoding of one individual into another while considering empty plies to blend with any orientation. Forced convergence of the umbra migration method creates global designs that are perfectly blended using the same measure of blendedness given in Chapter 5. The best discovered design had a total structural weight of 29.19 kg and falls within the domain of guide designs explored by the current work. In fact, all converged designs using umbra migration fall within this domain. It is interesting to note though that once convergence is achieved it becomes impossible for other blended designs to be evaluated by this method. The user has little confidence in the quality of the answer until

TABLE 9. *Other guide based designs found. Each design is described by three lines. Line 1: guide design. Line 2: constructor defining the number of plies for panels 1–18, left to right. Line 3: weight (kg)/fitness value (reflects constraint margins, larger is better).*

$[60/90_2/75/0/ \pm 45_2/30/ \mp 30/45/ \mp 45/ \pm 60_3/ - 75/60/75]_s$ 17 14 11 9 8 11 9 13 19 18 15 14 11 9 13 19 9 11 28.63/ – 81.103688
$[15/90/15/0/90/ - 45/ \pm 45_2/30/ \mp 30/ \pm 45/ \pm 60/75/ \pm 60/45/60/ \mp 45]_s$ 17 14 11 9 8 11 9 13 19 18 15 14 11 9 13 19 9 11 28.63/ – 81.173725
$[-60/90/ - 30/90/15/ \pm 45_2/30/ \mp 30/45/ \mp 45/60/ \mp 60_3/ \mp 75]_s$ 17 14 11 9 8 11 9 13 19 18 15 14 11 9 13 19 9 11 28.63/ – 81.106705
$[75/15/ - 30/90/ - 60/45/ \mp 45_2/30/ \mp 30/ \mp 45/60/ \mp 60_2/ \mp 75_2]_s$ 17 14 11 9 8 11 9 13 19 18 15 14 11 9 13 19 9 11 28.63/ – 81.093024
$[90/30/ - 45/ - 30/90/ \pm 45_2/ \pm 30/45/30/ \mp 45/60/ \mp 60_3/ \mp 75]_s$ 17 14 11 9 8 11 9 13 19 18 15 14 11 9 13 19 9 11 28.63/ – 81.106640
$[-75/45/60/0/90/45/ \mp 45_2/30/ \mp 30/ \mp 45/ \mp 60_2/75/ \mp 60/ \mp 75]_s$ 17 14 11 9 8 11 9 13 19 18 15 14 11 9 13 19 9 11 28.63/ – 81.091774

many runs have been made and, unlike other GA based methods, additional generations do not provide the possibility of finding a better solution on a single run.

Using guide based design even for a small number of generations, given the size of the problem design space, produced blended designs that were consistently better than the best known umbra migration results. Table 8 displays one of many optimal (best known) designs discovered with a weight of only 28.63 kg. In fact, approximately 90% of the runs conducted for 2000 iterations had a final design weight of 28.63 kg, differing only slightly in their fitness values, caused by different constraint margins (larger constraint margins are better). Not only does this provide the practicing engineer with a wealth of good designs to choose from, the algorithm itself is more robust (than those in [2] and [30]) and can continue to find possibly better solutions given more generations to explore the design space. Table 9 lists some of these other designs found, showing the variability possible among good designs. Perhaps the strongest argument for using GAs for composite structure design is the (typical) production of many dissimilar yet quite good composite designs.

## Chapter 7: SUMMARY FOR FIXED LOAD METHODS

A methodology for blending of laminate stacking sequences across multiple composite panels of a large composite structure using genetic algorithms in a parallel processing environment is proposed. The methodology relies on simultaneous evolution of the stacking sequences of the individual panels mapped to different processors that communicate with one another via migrant individuals. A metric based on Levenstein edit distance is used to evaluate the similarities of the stacking sequence between adjacent panels. Top individuals in the independently evolving populations are sent to neighboring populations in an effort to coerce the evolution of the neighboring panels so as to minimize the differences in laminate stacking sequences. Blending is encouraged by comparing each of the members of the neighboring populations with the migrant individual(s) and modifying their fitness based on the degree to which they match to the migrant. A scaling factor is used to adjust the degree of pressure imposed on blending of the neighboring panels. Two different communication topologies are used in distributing the top migrating individuals. In a ring topology, each panel (processor) communicated with the panel whose identification number followed it, with the last panel communicating with the first one. The second communication topology mirrored the physical configuration of the neighboring panels in the test problem.

Results are generated using the two different topologies with either single or multiple migrants, and by changing the scaling factor to investigate its influence on the blending characteristics of the neighboring laminates. Limited success was achieved in designing structures with completely matched laminates across panels. It was found that for those problems where the loading and neighborhood arrangements were such that the resulting laminate designs had inherent tendencies (such as decreasing laminate thickness from one side of the structure toward another side), using a communication topology that matched the tendency resulted in good blended designs when a single migrant was used. However, contrary to intuition, increasing the number of migrants had an adverse effect on the blending of the laminates, causing the GA to converge to (speciously) locally optimal laminates with sometimes severe mismatch between adjacent panels. It was further discovered that duplicating the exact topology of the neighboring panels in the communication topology

between the processors, the GA would sometimes converge to locally nonoptimal nonmatched designs even when a single migrant was used. By varying the scaling parameter, it was shown that a set of designs with varying degree of blending and weight compromise could be generated that can aid engineers in the design of large structures with multiple panels. However, for complex panel neighborhood configurations, the approach has to be used with caution.

A new guide based parallel GA has been proposed for composite panel structure optimization. For an 18-panel test problem, the present algorithm produced the lightest known design, as well as a variety of minimal weight near optimal designs. This production of several substantially different near optimal designs has practical importance, and is perhaps the strongest justification for the use of GAs in composite structure design. The definition of blending here amounts to perfect blending, although a less restrictive blending definition could still result in designs manufacturable with the minimal number of tow machine passes. Also, in general the local loads depend on the panel designs, so the algorithm here must be iterated to a local load fixed point.

## Chapter 8: COUPLED LOAD GUIDE BASED DESIGN EXTENSION

The initial approach to this problem involved the specification of a fixed load for each panel in the structure [2], [30], [20], [15]. Isolated local panel optimization results in stacking sequence orientations that vary widely between adjacent panels, which would cause serious manufacturing difficulties, generating the need for a globally blended solution. By fixing the local loads on each panel, the complexity of the global problem is greatly reduced, but the issue of blending the optimized local (panel) designs into a complete structure design that is manufacturable is far from trivial. Earlier work on this blending problem included a genetic algorithm (GA) method using the edit distance metric to allow a set of independently evolving panel populations to evolve to a blended global solution using reference migration [2], the use of sublaminates definitions and design variable zones by Soremekun et al. [30], and the addition of continuity constraints proposed by Liu and Haftka [17]. The edit distance method utilizes a multiple-population parallel GA approach that obtains blended designs through evolutionary pressures from neighboring populations. It is a nonstandard parallel decomposition that works asynchronously in real time to emulate semi-isolated populations with random migration to produce blended global designs from a pool of globally unconstrained local stacking sequence design possibilities. Evolutionary pressures are controlled through a user defined scaling factor that modifies the severity of penalties imposed for blending mismatches. These penalties, however, were found to hinder convergence to a global optimum by creating local optima in the search space that are artifacts of the algorithm itself [2]. The approach used in [30] is a two step procedure that relies on first optimizing the individual panels followed by identifying common thickness zones across multiple panels that are redefined and reoptimized using blended stacking sequences. Because of the heuristic nature of the approach, however, it is possible that suboptimal designs are generated, although blended designs are obtained with little weight penalty compared to unblended minimum weight structural design.

A more recent approach to this problem is a guide based design methodology [3]. In guide based design, the search space of the problem is implicitly reduced by forcing the generation of global (defining the entire structure) individuals that are always completely



blended across panel boundaries. The particular focus of the present work is in extending the guide based design methodology to a more realistic representation of the problem where the local loadings for individual panels can only be determined through a global level analysis that is coupled with the exact stacking sequence for each panel in the structure. This extension creates a challenging subproblem to determine the fitness of a guide design given the additional coupling of panel designs. A global/local iterative process is presented here that quickly converges to a near optimal solution and, with a small amount of subdigraph expansion (searching the subdigraph in some way until no valid paths are found), the optimal solution for a given guide can be found. The solution to this subproblem appears to be sufficiently fast that it can be incorporated into a GA for global optimization of the structure where each member of the population is an instance of the subproblem for which a fitness value can be determined.

Section 8.1 and 8.2 describes a global/local approach to the subproblem created by observing that a multi-panel wing structure is statically indeterminate. Changing the stacking sequence of one panel can change its stiffness properties, which affect the internal load distribution, possibly changing the in-plane load of every other panel in the structure. Section 8.3 supports the accuracy of the global/local approach through comparison to smaller problems with known optima discovered through exhaustive search. A parallel genetic algorithm is proposed to incorporate the global/local subproblem evaluation in a global optimization process and demonstrates the effectiveness of the method by finding several designs better than the best previously discovered in the literature. Chapter 9 presents a short lessons learned section about the approach and the high computational cost associated with it.

## 8.1 GUIDE BASED DESIGN

A simplified definition of blending is used in this work. Two adjacent panel designs are outwardly (inwardly) blended if one design is obtained by deleting a contiguous series of outermost (innermost) plies from the other. A global design is perfectly outwardly (inwardly) blended if and only if blending holds for every pair of adjacent panels starting from the

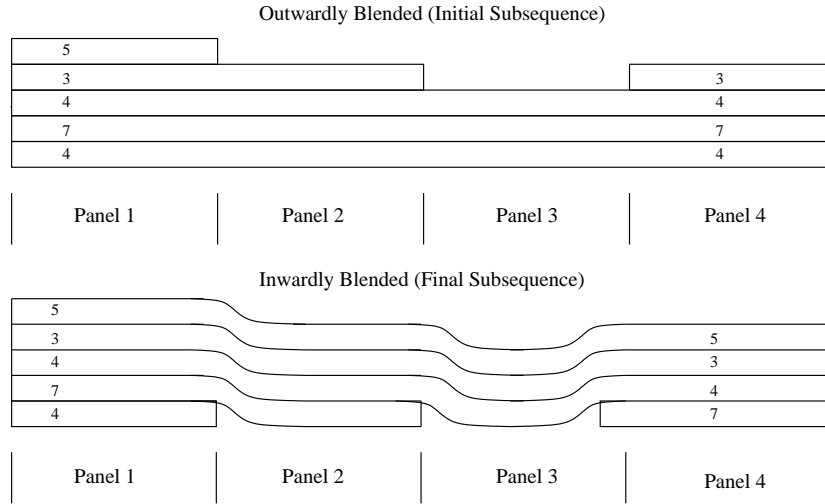


FIG. 8. *Outwardly/inwardly blended distinction example.*

innermost layer near the mid-plane (starting from the outermost layer and moving inward toward the mid-plane).

Using the simplified definition of blending, a perfectly outwardly (inwardly) blended global design can be generated by designing each local panel as an initial (final) sequence ( $a_1, \dots, a_k$  is an *initial sequence*, and  $a_{n-k+1}, \dots, a_n$  is a *final sequence*, of the sequence  $a_1, \dots, a_k, \dots, a_n$ ) of a single larger guide design. Figure 8 displays the distinction between outward and inward blending rules. A GA can be created to operate on and generate a population of individuals that guide the global design process so that only globally blended designs are in the problem domain. During the analysis phase of the GA a guide design (a single individual from the population) is evaluated to determine the optimal initial sequence for each local panel satisfying loading constraints and minimizing weight. This is accomplished by stripping ply layers from the guide design, starting from the outermost (for outward blending) layers, one layer at a time and analyzing the resulting designs according to the constraints for each local panel. (More efficient variations of this are possible, e.g., starting with the initial sequence length equal to that of the last optimal design analyzed for this panel, and then adding/subtracting layers from this length.) The guide designs in the population acquire fitness values through the combination of optimal local designs generated by the guide itself so that each guide design has a single best series of initial sequences corresponding to each panel that can be used to construct a usable fitness function

value. An optimal initial sequence is discovered for each local panel and the fitness of the global design becomes the sum of the individual fitness values for every optimized local panel. This modified analysis method provides the means to assign fitness values to guide designs themselves and allows a GA to operate on populations of guide designs to drive a global optimization process to an optimal blended design. An illustrative fictitious example of guide based design is presented in Figure 7.

## 8.2 GUIDE DESIGN SUBPROBLEM

Guide based design is important because it reduces the global search space by examining only those designs that are completely blended. The reason that it works well in the above context is that it is computationally cheap to discover the optimal design corresponding to any particular guide design. Since the loads on all panels are fixed, a one dimensional optimization performed on each panel independently from the other panels is sufficient to discover the lightest possible design that satisfies the loading constraints. For fixed loading, this design is unique and can be easily assigned a fitness by the controlling GA. Guide based design can be summarized as a methodology to construct blended designs from a given thick laminate, wrapping this process into a global optimization GA to search for the best possible guide design laminate in the space of possible designs.

The problem with the above method is that loads are not fixed locally. A fixed load at the tip of the wing will distribute that load to each panel according to the stiffness and strength properties of each panel in the entire structure. For a given guide laminate, the fitness value (to be maximized) for the guide is defined as

$$-(W)(P + 1),$$

where  $W$  is the total weight of the structure and  $P$  is the total penalty applied to the structure for being unbalanced. As stated in Section 3.1, the encoding for each laminate is a string of integers where successive occurrences of a gene map to the corresponding orientation with alternating plus and minus signs. A balanced stacking sequence contains an even number of each individual gene such that every occurrence of a positive orientation

angle has a complimentary negative orientation somewhere in the laminate. The value of  $P$  is the sum of the number of genes occurring an odd number of times in the stacking sequence for a panel summed across all panels in the structure. If, however, any of the panels in the structure fail under their local loading requirements an additional multiplicative penalty is applied to the fitness calculation making failed designs highly undesirable. The precise fitness value for a guide design is defined as the minimum weight design such that all local panel loading requirements are satisfied and in cases where multiple configurations exist at the minimum weight for a given guide, a design minimizing the value of  $P$  from among the equal configurations is selected to represent the fitness. Note that this definition of fitness for a guide may not represent the best possible fitness value attainable from a given guide since it favors weight before considering fitness value. This choice was made with the intention of steering the GA search toward designs using fewer layers even if they are not yet balanced.

The determination of the fitness of a guide is itself a complex optimization problem where the discovery of the optimal solution, as described above, may not be computationally feasible. The following sections describe methods used in this work to approximate the fitness value of an arbitrary guide.

### 8.2.1 GLOBAL/LOCAL ITERATION

In order to fit the guide based design approach to the problem where loads vary as the stacking sequences change a few changes need to be made. The following sections describe an algorithm with three distinct tiers with the overall goal of determining the fitness of a single guide. The top tier quickly finds a good starting point for a more fine grained search. The middle tier refines the search with a greedy algorithm and the bottom tier attempts to exhaustively search the remaining subdigraph of designs. The first proposed modification of the guide based design algorithm is to iterate between an exact global analysis and local level panel optimizations for specific loading conditions until convergence — a *global/local iteration*. The evaluation of a single guide begins with the global analysis of a completely thick design: each panel in the structure is assumed to use every possible layer from the

given guide. A completely thick design can be represented as a pair with a particular guide encoding and a string of zeroes corresponding to the number of ply layers removed from each panel in the structure:

$$[GuideEncoding]_s; [0, 0, \dots, 0].$$

Every configuration of values for ply layers removed, given in the second string above, represents a possible panel design configuration for the entire structure with corresponding local loadings on each panel. Since the determination of these local loads requires an expensive global finite element analysis, it is desirable to do as few global analyses as possible. The global analysis of the completely thick design yields local loading information for each panel in the structure used as input to the local level analyses to determine the failure status of each panel. Though the local loads given from the global analysis are only valid assuming the stacking sequences for each panel in the structure remain unchanged, these loads can allow for a local level linear optimization to guess a new design point to examine with a global analysis.

The local level optimizations take a single panel with the given loads returned from the most recent global optimization and determine the minimum length initial subsequence from the guide that can satisfy the given loading requirements. This operation can be performed independently on each panel and the local optimization results in a tuple of nonnegative integers corresponding to the number of ply layers removed from the guide for each panel to form the minimum weight structure possible under the given loads. For a given guide, this tuple fully describes an instance in the design space of all possible global designs derivable from the guide. This tuple becomes the best guess at which the next global analysis should be performed. The global analysis is required because by removing ply layers from the panels the stiffness properties change and the load distribution for the entire structure can change. The global analysis will yield new loading parameters for each panel in the structure and new linear local searches can be performed to determine the lightest possible structure based on the new loads. The process iterates between a global level analysis and the local level panel optimization under the following rules:

1) After a global analysis, the local loadings on all panels are exact until a change is made in the stacking sequence of a panel. If, under the given loads, a panel fails at the local level, then that panel must be made thicker and it is marked with a Boolean variable disallowing the removal of ply layers during future iterations. Continue iterating between the global level analysis and independent local optimizations.

2) If, after the global analysis, local optimization returns a tuple that satisfies local loading constraints for each panel and does not try to reduce the thickness of any remaining panels unmarked from Rule 1 then convergence is declared and the iteration terminates. Otherwise, continue iterating between the global level analysis and independent local optimizations.

The above finite process, summarized in Figure 10, produces a unique design for each guide obtainable in a very quick iterative process (generally less than five iterations). There is no reason to believe that this design is optimal, given the criteria for determining the fitness of a guide above, but assume (at least for now) that the unique design found through this global/local search is the source of a directed acyclic subgraph containing the optimal solution (optimal design derived from the given guide). Each guide design can be encoded as a string. A corresponding design for the structure is encoded as an array of integers representing the number of layers removed from the guide for each panel in the structure. This creates an acyclic directed graph structure (with a source node) of all possible designs (encodings) derivable from a given guide by manipulating the corresponding array of panel descriptors. The source for a four panel structure would be the fully thick design designated by the list  $(0,0,0,0)$ , where no ply layers are removed from the guide for any panel. The position of a number in the list corresponds to the design of a specific panel. The children of a specific node in the digraph correspond to all possible designs obtainable by removing a single ply layer from each panel in turn. For this example, the four children of the source node are the designs  $(1,0,0,0)$   $(0,1,0,0)$   $(0,0,1,0)$  and  $(0,0,0,1)$ . An illustrative example of a very small subdigraph for a 3 panel problem with a maximum thickness of 3 ply layers is given in Figure 9. An exhaustive search of the directed acyclic subgraph using the node corresponding to the unique design obtained through the global/local iterative approach as

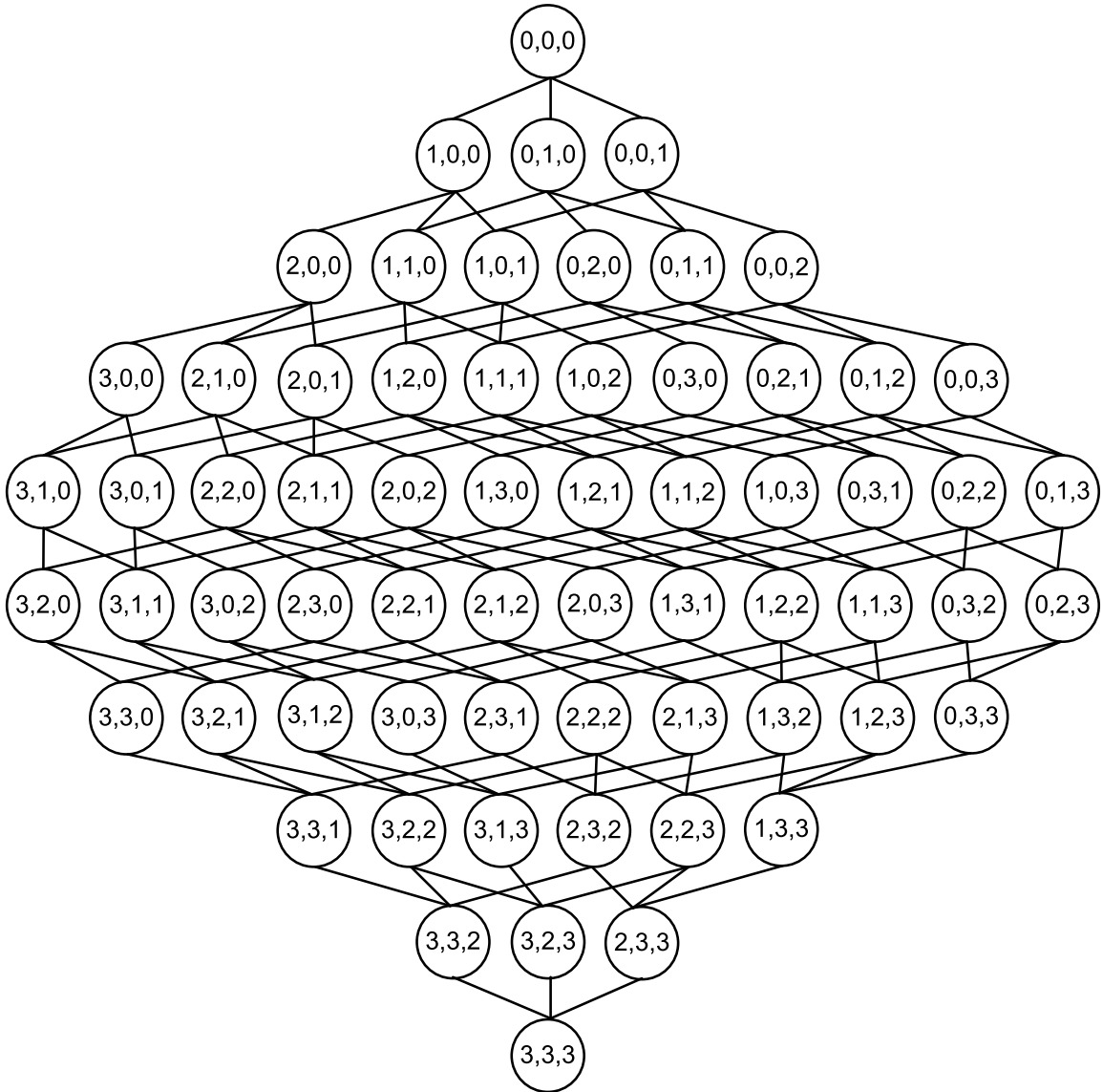


FIG. 9. *Sample 3 panel subdigraph using 3-ply thick laminates (all arcs directed downward).*

the source, stopping at nodes that correspond to infeasible designs, will discover the optimal solution under the following two assumptions:

1) the unique design discovered through this global/local iteration approach corresponds to the source node of a directed acyclic subgraph containing the (node describing the) optimal solution;

2) a directed path exists from the source node to any node describing a feasible design (under loading calculated by a global analysis) containing only nodes also describing feasible designs.





## 8.2.2 MIDDLE TIER ADDITION

For problems with few panels, the combination of the global/local iteration (top tier) and an exhaustive search of the directed subgraph (bottom tier) resulting from the converged design was computationally feasible. However, for larger structures it was quickly realized that the directed subgraph could contain hundreds of thousands of nodes for some guide configurations and the need for a middle tier search method was highlighted. The basic idea of the global/local iteration is to get a design that is close enough to the optimal solution that the latter can be found in a reasonable number of analyses of the exhaustive digraph search. The problem is that the global/local iteration, since it tries to optimize all panels in parallel at the local level, can converge at a node too far from the node corresponding to the fitness value of the guide. Consider, for example, a case where a structure with four panels is at the node

$$[12, 13, 14, 14]$$

in a particular global/local search iteration. When the node is analyzed with the global analysis, loads are determined for each panel corresponding to that particular node. It is possible that for those particular loads each panel, through local level analysis, decides that it can remove an additional layer without failing. The new node for examination then becomes

$$[13, 14, 15, 15].$$

When this node is evaluated with the global analysis, loads are again determined for each panel corresponding to this new configuration. If, for example, all four panels now fail at the local level, then the next node to be considered would again be  $[12, 13, 14, 14]$ , except this time all panels are marked with a failure flag and can not have further ply layers removed. If the optimal design happens to be  $[13, 14, 14, 15]$ , three levels below the source node, then the directed subgraph search would have to go three levels deep to find the fitness value of the guide. For a problem with only four panels this is not really an issue, but, for the eighteen panel problem compared later here, the same situation could cause the source node to be up to 17 levels above the optimal node corresponding to the fitness of the particular

guide in question. This would create a directed subgraph that would be far too large to exhaustively search.

The first middle tier modification is to create a computational budget for each guide evaluation. Any subdigraph search that takes more computational time than the allotted amount will need to be dealt with in a different way. This issue is combatted in three different ways, with the first being a middle tier greedy algorithm intended to force the source node further down in the guide graph structure. Instead of starting the exhaustive subdigraph search immediately, the source node is evaluated with its given loads and the constraint margins for all panels are sorted into a list. The greedy approach then tries to remove a ply layer from the panel with the greatest constraint margin. The resulting design is analyzed globally to create new local loading values and then examined for failure. If the design is valid then the new design has its local constraint margins sorted and the process starts again, but if it fails in some way, then the panel with the next highest constraint margin has a ply layer removed and that design is examined. This process continues until an iteration where the entire sorted list of panels has a single ply layer removed and all of them fail in some way. This design corresponds to the rock bottom of a greedy approach since no more layers can be removed from the design. An exhaustive search starting from this node would be pointless, but in order to find the most balanced node possible the computational budget allows for an exhaustive search of about four levels deep. This is accomplished by recording the last four moves of the greedy algorithm and backtracking up the digraph to that node, or to the node corresponding to the design resulting from the top tier algorithm, whichever is closer to the final greedy middle tier node. This node becomes the source node for the exhaustive search (bottom tier algorithm). Figure 11 summarizes the middle tier algorithm just described.

The greedy middle tier modification is sufficient for making most guide designs fall within the computational budget, but some outliers still produce large subdigraphs. Two additional rules were introduced to save computational time. If a guide design, after global/local iteration and the middle tier greedy search, still falls 10 levels or more above the best design found so far then it is highly unlikely that this guide is going to produce a

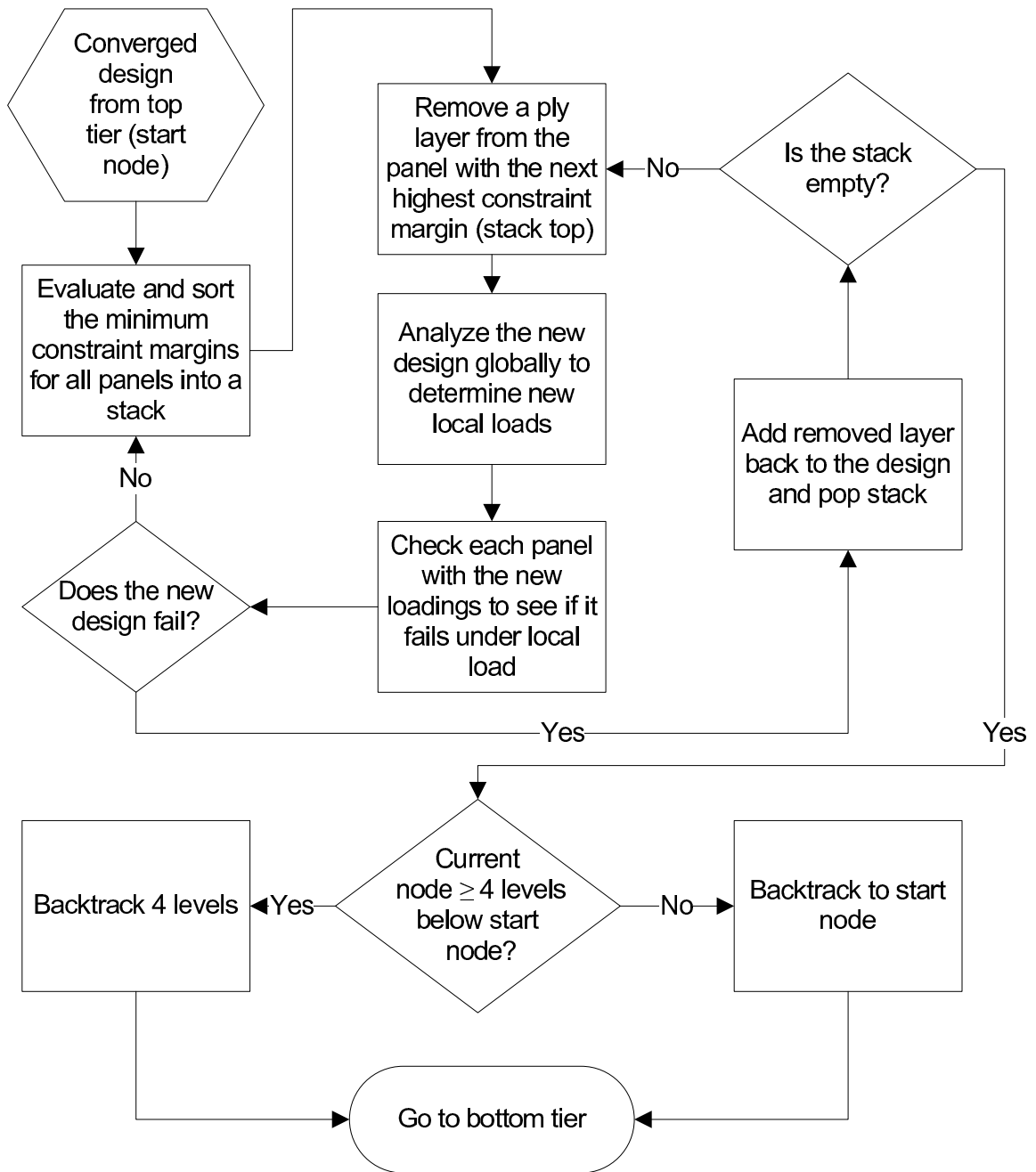


FIG. 11. Middle tier flow chart.

solution that is even competitive for the search. These designs are not even passed to the subdigraph search routine but rather return a fitness value corresponding to that given by the greedy approach and no more compute time is wasted. The second rule used is just a computational time kill switch placed in the subdigraph search routine. If a list of nodes to examine at a particular level of the subdigraph grows beyond a fixed length, 3600 for

the eighteen panel example given later or about an hour of compute time, then the search is aborted. The best fitness value discovered at that point in the search is reported as the fitness for the guide, and this guide is also reported in the output file as being relatively close to the best known solution but producing a subdigraph that was too large to search in the context of the larger optimization run. Most guide designs can be fully evaluated in less than two minutes (for the problems considered here) so cutting searches off that take more than an hour seems reasonable.

## 8.3 RESULTS

### 8.3.1 SIMPLE EXAMPLE

For the initial work, a simple wing structure model is used. The wing model is an unswept, untapered, rectangular wing box with dimensions  $354.33\text{ cm} \times 224.03\text{ cm} \times 38.1\text{ cm}$  ( $139.5\text{ in} \times 88.2\text{ in} \times 15\text{ in}$ ) (Figure 12) [27]. The spars and the ribs divide the top and bottom skin of the wing box into four panels of equal size. The top and bottom skin panels are modeled using membrane elements. Only the top skin panels are being optimized, all other sections are fixed to the design ( $[\pm 45_5 45]_s$ ). The upper panel arrangement is shown in Figure 13. The root of the wing is fixed and load is applied at the free end. Global finite element analysis is performed using the commercial finite element package GENESIS, developed by Vanderplaats Research & Development, Inc. The upward lift force acting on the wing tip is modeled by three concentrated loads of magnitudes  $3801.57\text{ nt}$  (at the leading edge wing tip),  $1878.79\text{ nt}$ , and  $1878.79\text{ nt}$  ( $854.67\text{ lbf}$ ,  $422.39\text{ lbf}$ , and  $422.39\text{ lbf}$ ).

Two guide designs were randomly selected and fixed for the top four panels in the wing and examined extensively. All possible structures derivable from the guides consisting of 160,000 design points each (nodes in the digraph) were explored exhaustively to determine which node should represent the fitness of the guide. In both cases the global/local method and subdigraph search described in Chapter 4 discovered the optimal solution for each subproblem in fewer than 27 global analyses.

A parallel GA, using the local/global method to evaluate guide designs, is constructed where the determination of fitness for a member of the population is assigned to a worker

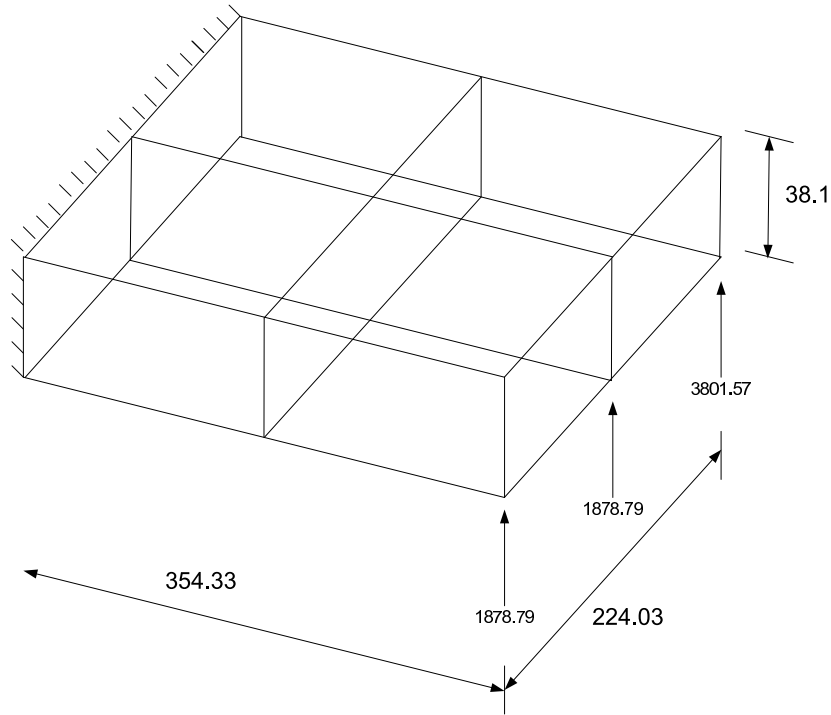


FIG. 12. *Wing structure model.*

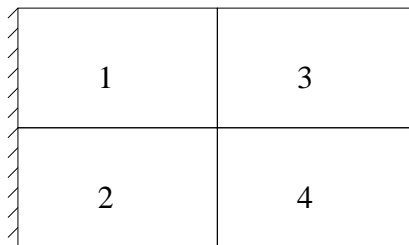


FIG. 13. *Panel layout for top and bottom skin panels.*

node during the analysis phase of each generation while the GA itself is controlled by a single master process. A static load distribution scheme is used in which each worker node receives an equal number of population members to analyze each generation. This is the same work distribution scheme used in the initial guide based design work by Adams et al. [3], but in this case the amount of computation required to determine the fitness of a single member of the population is not static. The variability in analysis times between individuals creates some load imbalances that will need to be corrected dynamically.

All testing for this problem was performed on a 200 node (400 processor) 1.4 GHz dual Opteron cluster using an MPI (message passing interface) based Fortran 90 code. However,

the small problem size and the fact that the determination of fitness of an individual is viewed as an atomic operation prevented the use of more than 51 processors at a time. The average run time for a GA using a population size of 50 running for 200 generations was about 12 hours (612 node hours). Initial results are promising. As a point of comparison, the same problem above was examined using the algorithm by Seresta et al. [27]. For a run of 200 generations of the GA, Seresta's algorithm discovered in 50 generations a solution that it could not improve after an additional 150 generations. The guide for this design is given as the encoding

$$(4, 7, 1, 4, 4, 7, 4, 4, 4, 7, 1, 1, 4, 4, 1, 1, 7, 4, 1, 4),$$

using the final ten layers of the guide for Panel 1, 11 layers for Panel 2, six layers for Panel 3, and eight layers for Panel 4, yielding a combined mass of 12.371 *kg* (27.274 *lbs*). The greedy global/local with bounded implicit enumeration (GGLBIE) algorithm presented here discovered a different solution of equal mass in one to four generations of the GA and a lighter design was discovered after 100 iterations. This lighter design is given as the guide encoding

$$(7, 4, 7, 4, 4, 7, 7, 4, 1, 7, 4, 4, 1, 1, 1, 4, 1, 4, 1, 7),$$

using the final ten layers of the guide for Panel 1, 11 layers for Panel 2, six layers for Panel 3, and seven layers for Panel 4, yielding a combined mass of 12.018 *kg* (26.494 *lbs*).

### 8.3.2 LOAD BALANCING MECHANISM

Since the time to analyze a single guide can take between two minutes and multiple days, a load balancing scheme had to be introduced. Part of the load balancing scheme is reflected in the computational budget allocated to each analysis. Any analysis that expands to more than 3600 nodes to analyze during the subdigraph search is reported, with the nodes beyond the first 3600 being discarded. The initial distribution scheme was to use 400 processors and a population size of 400. Each processor was given a single guide to analyze each iteration, but for most iterations this resulted in a single processor computing for about an hour and the other processors remaining idle, as they finished the work very quickly. The efficiency

suffered in the determination of fitness yielding about 0.2 guides per second on average. Using the same 400 processors and a population size of 400, a cut off point was introduced for all designs that were ten or more levels above the best known design after the middle tier greedy algorithm. The cut off point is just a mechanism for recognizing searches that most likely will not be competitive with the best known solution for the problem so far and even if they were to be competitive, their subdigraph searches would be too large to realize their potential. This process was accomplished by passing the information about the best known design found so far along with each unit of work and was updated throughout the optimization run even within a single generation. This increased the average number of guides analyzed to 0.8 guides per second. The major load balancing change though was in an increased population size and a modified work distribution scheme for the master-slave algorithm. In previous work [3] the master process would distribute work in equal sized chunks when the population size was larger than the number of worker processors available. This scheme was modified so that each worker processor would request a single guide to evaluate at a time and request new work when it finished the work it was given. At the same time the population size was increased by a factor of 50 to 20,000 individuals per generation. Using this basic load balancing scheme and a higher population allowed for the guides falling into the category of outliers to be analyzed for the entire duration of a generation while other workers could process many more routine guide evaluations sequentially. This method allowed the average number of guides analyzed per second to reach 9.5 and was sufficient for the following runs.

### 8.3.3 COMPARISON WITH OTHER PUBLISHED WORK

A previous section displayed a success story for a toy problem, but a larger test problem is required to compare the GGLBIE algorithm with other methods published in the literature. This section describes the results and difficulties in dealing with an eighteen panel problem originally studied by Liu and Haftka [17] and again by Seresta et al. [27]. It should be stated up front that this method requires significantly more computational power than the above mentioned published works, but is a more effective search mechanism. The trade off

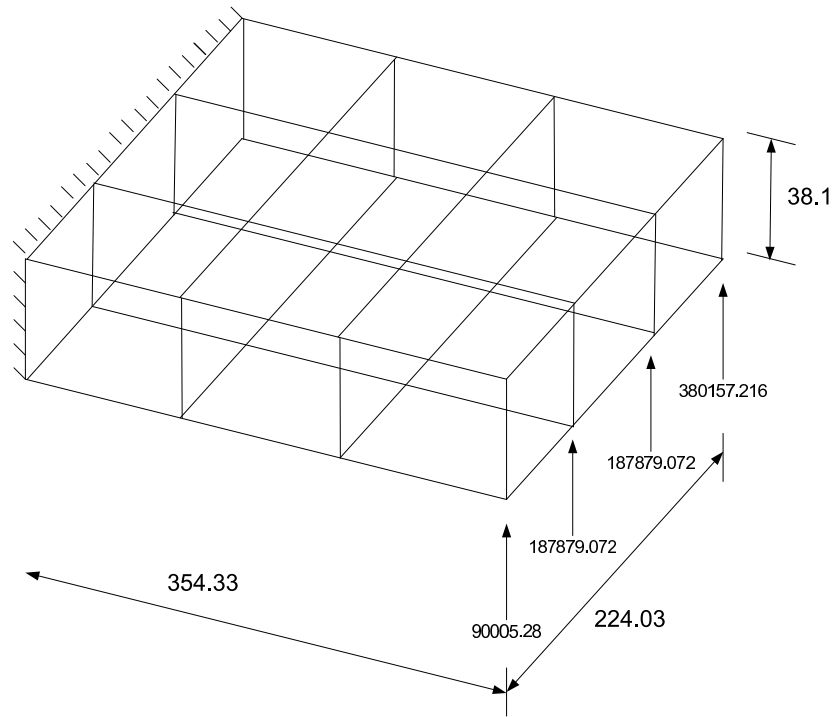


FIG. 14. 18 panel wing structure model.

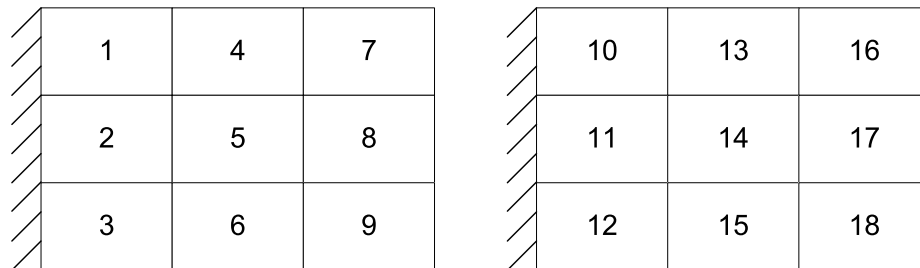


FIG. 15. Panel layout for top (left) and bottom (right) skin panels for the 18 panel problem.

for improved design quality in exchange for additional computational time can be weighed by the developer to evaluate the cost/benefit ratio for this method.

All testing for this larger problem was performed at Virginia Polytechnic Institute and State University on System X, a 1100 node (2200 processor) 2.3 GHz PowerPC cluster using an MPI based Fortran 90 code. A simple wing structure model is again used. The wing model is an unswept, untapered, rectangular wing box with dimensions  $354.33 \text{ cm} \times 224.03 \text{ cm} \times 38.1 \text{ cm}$  ( $139.5 \text{ in} \times 88.2 \text{ in} \times 15 \text{ in}$ ) (Figure 14). The spars and the ribs divide the top and bottom skin of the wing box into nine panels of equal size. The top and bottom



skin panels are modeled using membrane elements. The root of the wing is fixed and load is applied at the free end. The upward lift force acting on the wing tip is modeled by four concentrated loads of magnitudes 380157.216 *nt*, 187879.072 *nt*, 187879.072 *nt*, and 90005.28 *nt* (85467 *lbf*, 42239 *lbf*, 42239 *lbf*, and 20235 *lbf*) (Figure 14). The inwardly blended design reported in Table 10 and 11 was found after 94 iterations of the GA using a population size of 20000 and consumed about 140,000 node hours of compute time on 400 processors. The tables illustrate that all panels share the ply layers  $\alpha$  or  $\beta$  and should be read such that the entry

$$\begin{array}{|c|} \hline 4 \leftarrow 3 \\ \hline 2 \leftarrow 1 \\ \hline \end{array}$$

corresponds to the stacking sequence  $[1 \rightarrow 2 \ 3 \rightarrow 4]_s$  as described in Section 3.1. The best known solution for this problem was reported by Seresta [27] and had a mass of 255.91 *kg* (564.17 *lb*). The solution discovered by the GGLBIE algorithm had a mass of 252.73 *kg* (557.16 *lb*) using the inwardly blended approach, which the careful reader will determine is called outer blending in [27]. For reference, Table 12 gives the number of ply layers used for each panel and the corresponding mass and Figure 16 gives a topology mapped representation of the design indicating the thicknesses of each panel with a panel number reference above.

TABLE 10. *Stacking sequence of top panels ordered from thickest to thinnest.*

3	90	0	90	90	0	0	90	90	0	90	90	0	90	90	0	90	0	90	90	
	0	0	0	-45	45	0	0	90	0	0	90	0	0	0	90	0	90	0	90	$\alpha$
2						0	0	90	90	0	90	90	0	90	90	0	90	0	90	90
	0	0	0	-45	45	0	0	90	0	0	90	0	0	0	90	0	90	0	90	$\alpha$
1								90	0	90	90	0	90	90	0	90	0	90	90	
	0	0	0	-45	45	0	0	90	0	0	90	0	0	0	90	0	90	0	90	$\alpha$
6																0	90	0	90	90
	0	0	0	-45	45	0	0	90	0	0	90	0	0	0	90	0	90	0	90	$\alpha$
5																				90
	0	0	0	-45	45	0	0	90	0	0	90	0	0	0	90	0	90	0	90	$\alpha$
4																				
				-45	45	0	0	90	0	0	90	0	0	0	90	0	90	0	90	$\alpha$
9																				
									0	90	0	0	0	90	0	90	0	90	0	$\alpha$
8																				
																0	90	0	90	$\alpha$
7																				
																				$\alpha$
$\alpha$																				
	45	-45	45	-45	45	-45	90	0	45	0	90	-45	0	0	0	90	0	0	45	0

TABLE 11. *Stacking sequence of bottom panels ordered from thickest to thinnest.*

16	90	-45	45	90	90	0	0	90	0	90	90	0	90	90	0	90	90	$\beta$	
13				90	90	0	0	90	0	90	90	0	90	90	0	90	90	$\beta$	
14										90	90	0	90	90	0	90	90	$\beta$	
17										90	90	0	90	90	0	90	90	$\beta$	
10											0	90	90	0	90	90	$\beta$		
11												90	90	0	90	90	$\beta$		
18																90	$\beta$		
12																	$\beta$		
15																	$\beta$		
$\beta$																		0	
	0	90	0	0	0	0	90	90	90	90	0	90	0	90	90	0	90	90	90

TABLE 12. *Panel thickness and mass.*

Panel No.	Ply Layers	Mass <i>kg (lb)</i>
1	126	22.27 (49.09)
2	132	23.33 (51.43)
3	140	24.74 (54.55)
4	96	16.96 (37.40)
5	104	18.38 (40.52)
6	112	19.80 (43.64)
7	64	11.31 (24.94)
8	76	13.43 (29.61)
9	84	14.85 (32.73)
10	54	9.54 (21.04)
11	52	9.19 (20.26)
12	42	7.42 (16.36)
13	70	12.37 (27.27)
14	58	10.25 (22.60)
15	42	7.42 (16.36)
16	76	13.43 (29.61)
17	58	10.25 (22.60)
18	44	7.77 (17.14)

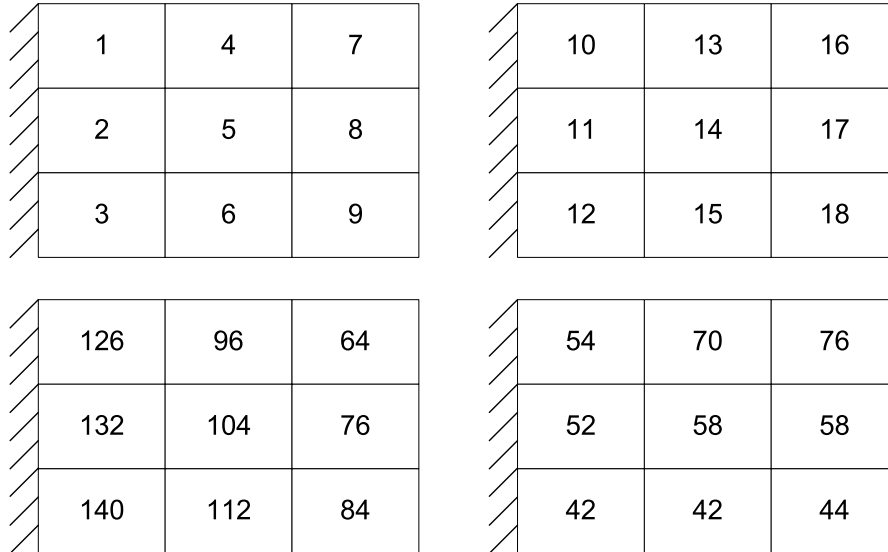


FIG. 16. *Panel thicknesses in topology mapped representation.*

## Chapter 9: CONCLUSIONS

The first observation about genetic algorithms for blended composite laminate structure design is that some restriction of the design space (e.g., a guide based GA considers only perfectly globally blended designs) seems necessary to make significant progress towards optimality. Simply because of the number of different fairly good blended designs that exist, many different heuristic approaches ([2], [30], [20], [15], [17], [3]) are able to find fairly good designs with moderate computational costs. None of these approaches, however, as the results here show, found a solution within 1% of optimal. The second observation is that, consistent with the prevailing wisdom in the evolutionary computation community, considerable effort to improve, refine, or “tune” individuals within a generation is cost effective. Here the three-tier deterministic algorithm, mixing greedy and implicit enumeration components, can be construed as a form of local improvement applied simultaneously to all the blended designs associated with a guide.

The reality of this work is that this type of combinatorial optimization problem is very difficult to solve efficiently. There may be ways to find a decent solution in a relatively short amount of time using other methods simply because there are a lot of relatively good designs in the design space, but to increase the quality of a solution toward the optimal for this type of problem requires a much larger expenditure of computing resources. This is typical of combinatorial optimization problems, and is true also for the traveling salesman problem, for example.

To summarize this body of work a reader should keep in mind that the first method, the edit distance method, was a stepping stone to get to the guide based design methodology. The edit distance method itself displayed algorithmic properties that were undesirable for optimization methods including premature convergence to false local optima. These local optima are called false because they are locally optimal only according to the blending measures evolutionary pressures and are an artifact of the algorithms construction. Even though the edit distance method was successful in generating results better than the best known results at the time, it is not suggested that work continue in this area. Rather, the lesson learned from this work was that a reduction of the design space was needed. The

relatively successful umbra migration modification to the edit distance method displayed the best properties of all the different communication topologies attempted. All designs that converge to a design with zero modified edit distance through umbra migration have the property that every panel encoding is a contiguous substring (initial in this case) of the single largest (most thick) encoding in the structure. This is the primary idea behind guide based design. The only real success found using the edit distance method toward perfectly blended designs was found through umbra migration, and guide based design is an attempt to exploit and explore those types of solutions.

The fixed load guide based design method provides a good foundation for future work. This method is the most successful of the three explored in this dissertation since it is fast, has trivial uniqueness proofs for the fitness of any given guide, and derives its convergence properties directly from the underlying GA parameters without interference from algorithm modifications. The guide based design portion of the algorithm is independent of the controlling GA such that the outer optimization control structure can be swapped out to any other fitness based optimization procedure. Any further study on this problem should first consider improvements to the basic fixed load structure guide based design process. It would be interesting to see how other optimization methods compare to the GA in finding near optimal guides for a given problem.

It has been observed that the guide based design method, when considering coupled loads with each stacking sequence change, can be very computationally expensive. Though it is the most realistic of the three methods proposed in this work, it may have limited applicability for real world problems. The guide based design method with fixed load assumptions for each panel appears to be the approach to continue exploring. Other future work should include an iterative method that runs the fixed load guide based design method on a problem until a good design is found. This new design can be analyzed to determine new fixed loads and another run of the guide based design GA can be applied. This process would more closely resemble a process already used by industry, but would have the ability to vary the panel design thicknesses. There is a need to explore the range of load changes on particular panels as stacking sequences change so that reasonable bounds can be placed on the fixed load algorithm. Such iterative processes, however, are known to diverge or converge to points that are not even locally optimal. Ultimately a theory similar to that of Haftka and Watson [10] needs to be developed for blended design.

## BIBLIOGRAPHY

- [1] D. B. Adams, S. Setoodeh, Z. Gürdal, L. T. Watson, *Pipeline Implementation of Cellular Automata for Composite Wing Design on Massively Parallel Multiprocessors*, Mathematical and Computer Modeling, to appear.
- [2] D. B. Adams, L. T. Watson, and Z. Gürdal, *Optimization and blending of composite laminates using genetic algorithms with migration*, Mechanics of Advanced Materials and Structures, 10 (2003), pp. 183–203.
- [3] D. B. Adams, L. T. Watson, Z. Gürdal, C. M. Anderson-Cook, *Genetic algorithm optimization and blending of composite laminates by locally reducing laminate thickness*, Advances in Engineering Software, 35 (2004), pp. 35–43.
- [4] D. B. Adams, L. T. Watson, O. Seresta, Z. Gürdal, *Global/Local Iteration for Blended Composite Laminate Structure Optimization Subproblems*, Proc. High Performance Computing Symposium 2005, M. Parashar and L. Watson (eds.), Soc. for Modeling and Simulation Internat., San Diego, CA, 2005, pp. 151–157.
- [5] T. Bäck, *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, NY, 1996.
- [6] C. Bisagni, L. Lanzi, *Post-buckling optimization of composite stiffened panels using neural networks*, Composite Structures, 58 (2002), pp. 237–247.
- [7] E. Cantú-Paz, *Efficient and Accurate Parallel Genetic Algorithms*, Kluwer Academic Publishers, Boston, MA, 2001.
- [8] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley, Reading, MA, 1989.
- [9] D. Gusfield, *Algorithms on Strings, Trees, and Sequences*, Cambridge University Press, New York, NY, 1997.
- [10] R. T. Haftka and L. T. Watson, *Multidisciplinary design optimization with quasiseparable subsystems*, Optim. Engrg., 6 (2005), pp. 9–20.
- [11] J. H. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, MI, 1975.
- [12] K. A. De Jong, *An analysis of the behavior of a class of genetic adaptive systems*, Ph.D. dissertation, Univ. of Michigan, Ann Arbor, MI, 1975.
- [13] T. Kim, H. Sin, *Optimal design of composite laminated plates with the discreteness in ply angles and uncertainty in material properties considered*, Computers and Structures, 79 (2001), pp.2501–2509.
- [14] N. Kogiso, L. T. Watson, Z. Gürdal, R. T. Haftka, and S. Nagendra, *Design of composite laminates by a genetic algorithm with memory*, Mechanics of Composite Materials and Structures, 1 (1994), pp. 95–117.
- [15] B. P. Kristinsdottir, Z. B. Zabinsky, M. E. Tuttle, and S. Neogi, *Optimal design of large composite panels with varying loads*, Composite Structures, 51 (2001), pp. 93–102.
- [16] V. I. Levenstein, *Binary codes capable of correcting insertions and reversals*, Sov. Phys. Dokl., 10 (1966), pp. 707–710.
- [17] B. Liu and R. T. Haftka, *Composite wing structural design optimization with continuity constraints*, in Proceedings of 42nd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, AIAA Paper 2001-1205, Seattle, WA, 2001.
- [18] B. Liu, R. T. Haftka, M. A. Akgün, A. Todoroki, *Permutation genetic algorithm for stacking sequence design of composite laminates*, Comput. Methods Appl. Mech. Engrg., 186 (2000), pp. 357–372.
- [19] M. T. McMahan and L. T. Watson, *A distributed genetic algorithm with migration for the design of composite laminate structures*, Technical Report TR-98-20, Dept. of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA, 1998.

- [20] M. T. McMahon and L. T. Watson, *A distributed genetic algorithm with migration for the design of composite laminate structures*, *Parallel Algorithms and Applications*, 14 (2000), pp. 329–362.
- [21] A. Muc, W. Gurba, *Genetic algorithms and finite element analysis in optimization of composite structures*, *Composite Structures*, 54 (2001) , pp. 275–281.
- [22] J. H. Park, J. H. Hwang, C.S. Lee, W. Hwang, *Stacking sequence design of composite laminates for maximum strength using genetic algorithms*, *Composite Structures*, 52 (2001), pp. 217–231.
- [23] M. J. Quinn, *Designing Efficient Algorithms for Parallel Computers*, McGraw-Hill Book Company, New York, NY, 1987.
- [24] R. Le Riche, *Optimization of composite structures by genetic algorithms*, Ph.D. dissertation, Department of Aerospace Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA 1994.
- [25] J. F. Rodríguez, J. E. Renaud, and L. T. Watson, *Trust region augmented Lagrangian methods for sequential response surface approximation and optimization*, *ASME J. Mech. Design*, 15 (1998), pp. 58–66.
- [26] J. F. Rodríguez, J. E. Renaud, and L. T. Watson, *Convergence of trust region augmented Lagrangian methods using variable fidelity approximation data*, *Structural Optim.* 15 (1998), pp. 141–156.
- [27] O. Seresta, Z. Gürdal, D. B. Adams, L. T. Watson, *Optimal design of composite wing structures with blended laminates*, 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference Albany, NY, Aug. 30–Sep. 1, 2004.
- [28] J. Shankar, C. J. Ribbens, R. T. Haftka, and L. T. Watson, *Computational study of a nonhierarchical decomposition algorithm*, *Comput. Optim. Appl.*, 2 (1993), pp. 273–293.
- [29] G. Soremekun, Z. Gürdal, R. T. Haftka, and L. T. Watson, *Composite laminate design optimization by genetic algorithm with generalized elitist selection*, *Computers and Structures*, 79 (2001), pp. 131–144.
- [30] G. Sormekun, Z. Gürdal, C. Kassapoglou, and D. Toni, *Stacking sequence blending of multiple composite laminates using genetic algorithms*, *Composite Structures*, 56 (2002), pp. 53–62.
- [31] R. Spallino, S. Rizzo, *Multi-objective discrete optimization of laminated structures*, *Mechanics Research Communications*, 29 (2002), pp. 17–25.
- [32] A. Todoroki, R. T. Haftka, *Stacking sequence optimization by a genetic algorithm with a new recessive gene like repair strategy*, *Composites*, 29B (1998), pp 277–285.
- [33] A. Todoroki, Y. Terada, *Stacking sequence optimizations using fractal branch and bound method*, in *Proceedings of 43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, AIAA Paper 2002-1298, Denver, CO, 2002.

## VITA

David Bruce Adams was born in Whitesburg, Kentucky on November 16, 1975. He earned a bachelors degree in Mathematics and Computer Information Systems in December of 1997 from what was then Clinch Valley College of the University of Virginia and is now the University of Virginia's College at Wise. The requirements for Master of Science in Computer Science from Virginia Polytechnic Institute and State University were satisfied in August 2002. He continued his studies at Virginia Tech in pursuit of the Doctor of Philosophy degree in Computer Science and finished in August 2005. Dr. Adams begins his teaching career at Grove City College in Pennsylvania on August 24, 2005.