Optimization and Blending of Composite Laminates Using Genetic Algorithms With Migration *

David B. Adams† Layne T. Watson‡ Zafer Gürdal§

Abstract

Optimization of aircraft stiffened composite panel structures often results in manufacturing incompatibilities between adjacent panels. 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.

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 [1], [2], [3]. This work involves the design of structures, with multiple composite laminates connected together to form the wing or fuselage of an aircraft.

Design of a fiber-reinforced composite laminate 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 of the present work 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. [3] introduced multiple elitist selection

* This work was supported in part by AFOSR Grant F496320-99-1-0128 and NSF Grant DMI-9979711.

† Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061-0106.

‡ Departments of Computer Science and Mathematics, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061-0106.

§ Departments of Aerospace and Ocean Engineering, and Engineering Science and Mechanics, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061-0203.

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 particular focus of this work is in 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 [1], 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 balancing the optimization of weight with stacking sequence compatibility.

Section 2 provides some background on the work in composite laminate design using genetic algorithms. Section 3 introduces the proposed blending algorithm providing pseudocode for a serial genetic algorithm and the direct parallel implementation with migration. Section 4 presents results for an 18 panel design problem in multiple communication topologies with varying parameters applied to the blending process.

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. 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 [4] in 1975, with application work following soon after in static function optimization by DeJong [5]. Goldberg [6] 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 [7] 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.

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 [8] introduced the concept of a two-stack swap

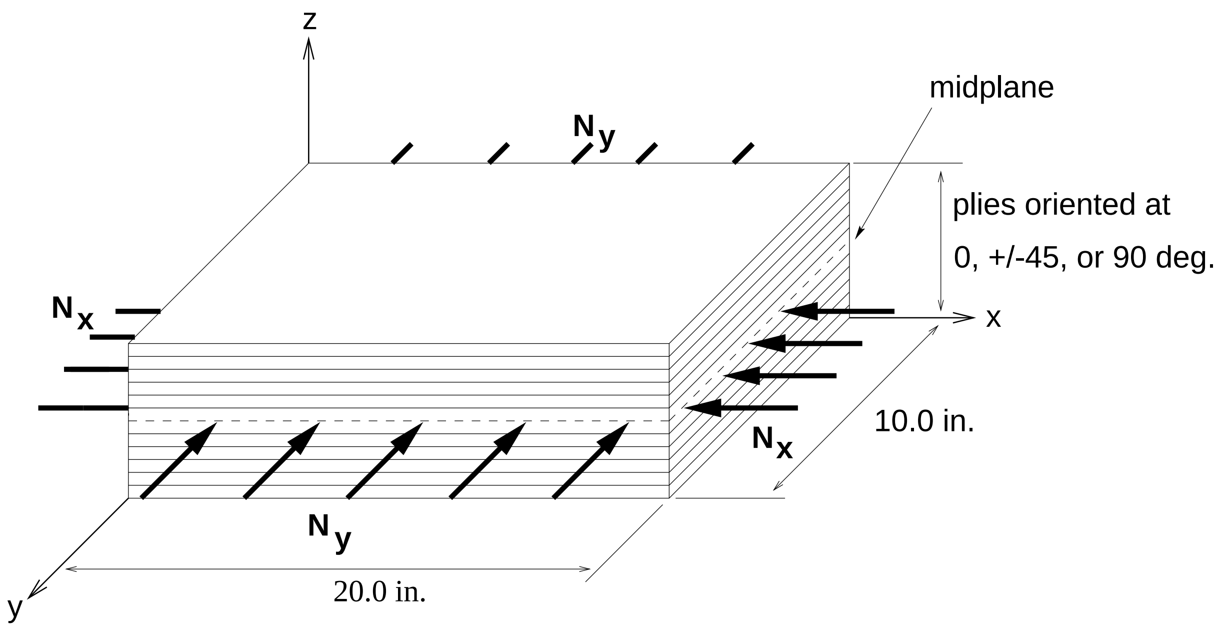

FIG. 1. *Local optimization problem for laminate stacking sequence.*

operator, which is generalized in the permutation and swap operators of McMahon [9]. McMahon 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. [3] are used in this work. Brief descriptions follow. The addition and subtraction 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 section.

3 The Engineering 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 [2], [1], [3]. 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 [10].

The 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 [9]. Using the same approach implemented by Soremekun [11], 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 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 [9], 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°, the second 2 maps to $-15°$, 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 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.

$$\begin{array}{rl} \text{chromosome before addition} & [1\ 2\ 3\ 6\ 2\ 1\ 2\ 4|2\ 5\ 7\ 3\ 3\ 1\ 0\ 0] \\ \text{chromosome after addition} & [1\ 2\ 3\ 6\ 2\ 1\ 2\ 4\ 1\ 2\ 5\ 7\ 3\ 3\ 1\ 0] \end{array}$$

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 $\underline{3}$ 6 2 1 2 $\underline{4}$ 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 ith best individual in a local population of interbreeding individuals or deme 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 paper 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 of this paper 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 populations are required to be unique with respect to the current filling child population and the generating parent 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 Section 4.

3.3 Serial GA Pseudo Code

For reference, a basic serial genetic algorithm is given here. Migration can be done serially, 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 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 [12], 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 [13].

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 [13]. 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 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 α, the scaling factor, is a user-defined constant that dictates the influence of blending on the optimization run. If the scaling factor α 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,

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

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

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

3.5 Parallel GA Pseudo Code

The pseudo code in this section outlines the actions taken by each processing node in the parallel environment. 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.

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 paper are less than one percent of total migrations.

This method is a variation on the migration procedure of McMahon [1] 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 Section 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
        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)
```

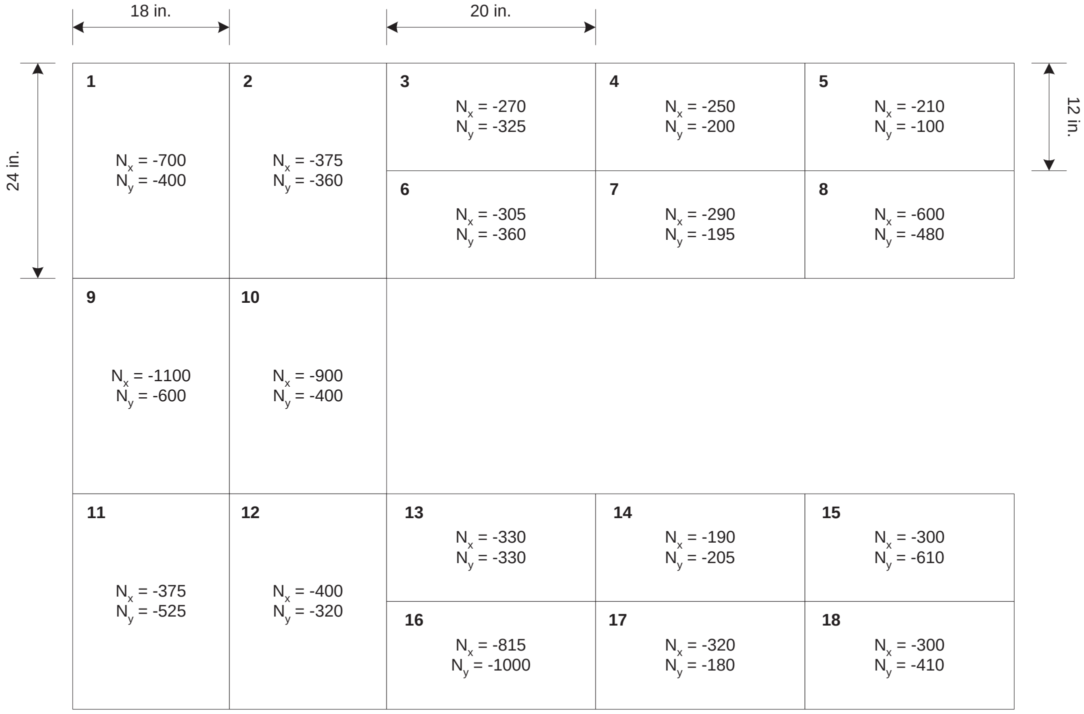

FIG. 2. *Eighteen panel irregular grid test problem.*

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

4 Results

Results are given here for a test problem consisting of eighteen panels configured and loaded as shown in Figure 2 [11]. The panels are composed of 0, ± 15, ± 30, ± 45, ± 60, ± 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 section.

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 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. Both methods indicate that the unblended design would be extremely difficult and expensive if not impossible to manufacture.

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 (*wait_iterations*)	4

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	$[75/\pm 75_5]_s$

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 α 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 increases the impact of the blending measure increases. At $\alpha = 8$ the structure converged to a perfectly blended design, using the standard edit distance metric, of

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

Though the structure has a total edit distance of zero for both manufacturability measures 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 many test runs using α 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 of design A and B, none of the panels can find a more blended design than they already 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

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	$-75/\mp75/\mp30/90/75/30]_s$
2	2.45	[		30	±30	-30	75	±75	$-75/\mp75/\mp30/90/75/30]_s$
3	1.03	[					75	±75	$-75/\mp75/\mp30/90/75/30]_s$
4	0.84	[						75	$-75/\mp75/\mp30/90/75/30]_s$
5	0.74	[							$-75/\mp75/\mp30/90/75/30]_s$
6	1.03	[					75	±75	$-75/\mp75/\mp30/90/75/30]_s$
7	0.93	[						±75	$-75/\mp75/\mp30/90/75/30]_s$
8	1.12	[				60	75	±75	$-75/\mp75/\mp30/90/75/30]_s$
9	3.35	[	$\pm30_2$	±30	±30	60	75	±75	$-75/\mp75/\mp30/90/75/30]_s$
10	3.02	[	±30	±30	±30	60	75	±75	$-75/\mp75/\mp30/90/75/30]_s$
11	2.68	[		±30	±30	60	75	±75	$-75/\mp75/\mp30/90/75/30]_s$
12	2.51	[		-30	±30	60	75	±75	$-75/\mp75/\mp30/90/75/30]_s$
13	1.03	[					75	±75	$-75/\mp75/\mp30/90/75/30]_s$
14	0.84	[						75	$-75/\mp75/\mp30/90/75/30]_s$
15	1.11	[				-75	75	±75	$-75/\mp75/\mp30/90/75/30]_s$
16	1.67	[	±75	±75	±75	-75	75	±75	$-75/\mp75/\mp30/90/75/30]_s$
17	0.93	[						±75	$-75/\mp75/\mp30/90/75/30]_s$
18	1.03	[					75	±75	$-75/\mp75/\mp30/90/75/30]_s$

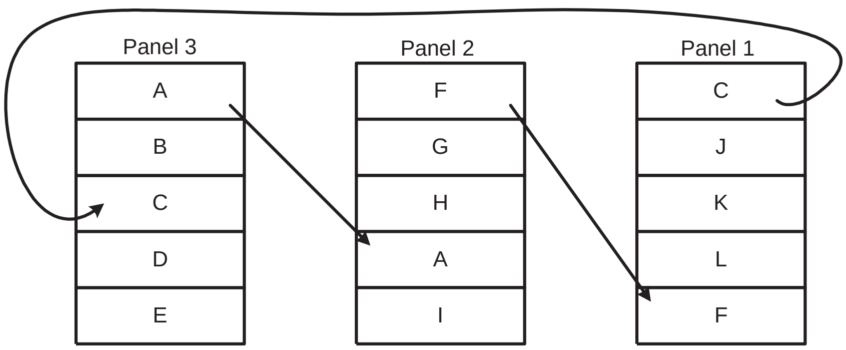

FIG. 3. *Migrant box of size five, for ring* $1 \to 2 \to 3 \to 1$. *The top five designs of each panel become its emigrants.*

converged locally 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 α large, the algorithm is forced to try and perfectly match the panels as any global design with a

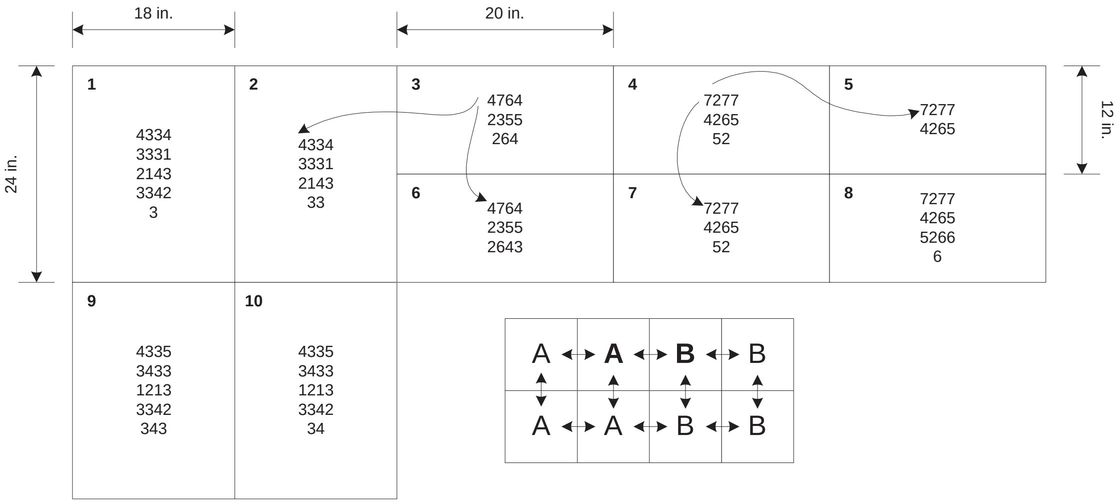

FIG. 4. *False optimum of topology mapped communication. Each panel shows the final (best) converged design, written in groups of four encodings.*

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 Section 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 [14]. Nevertheless, paradigms do exist wherein independent subsystem optimizations can be guided to an overall optimal system design (see, e.g., [15], [16]). Using these other paradigms will require a major reformulation of the present blending problem—this remains a topic for future research.

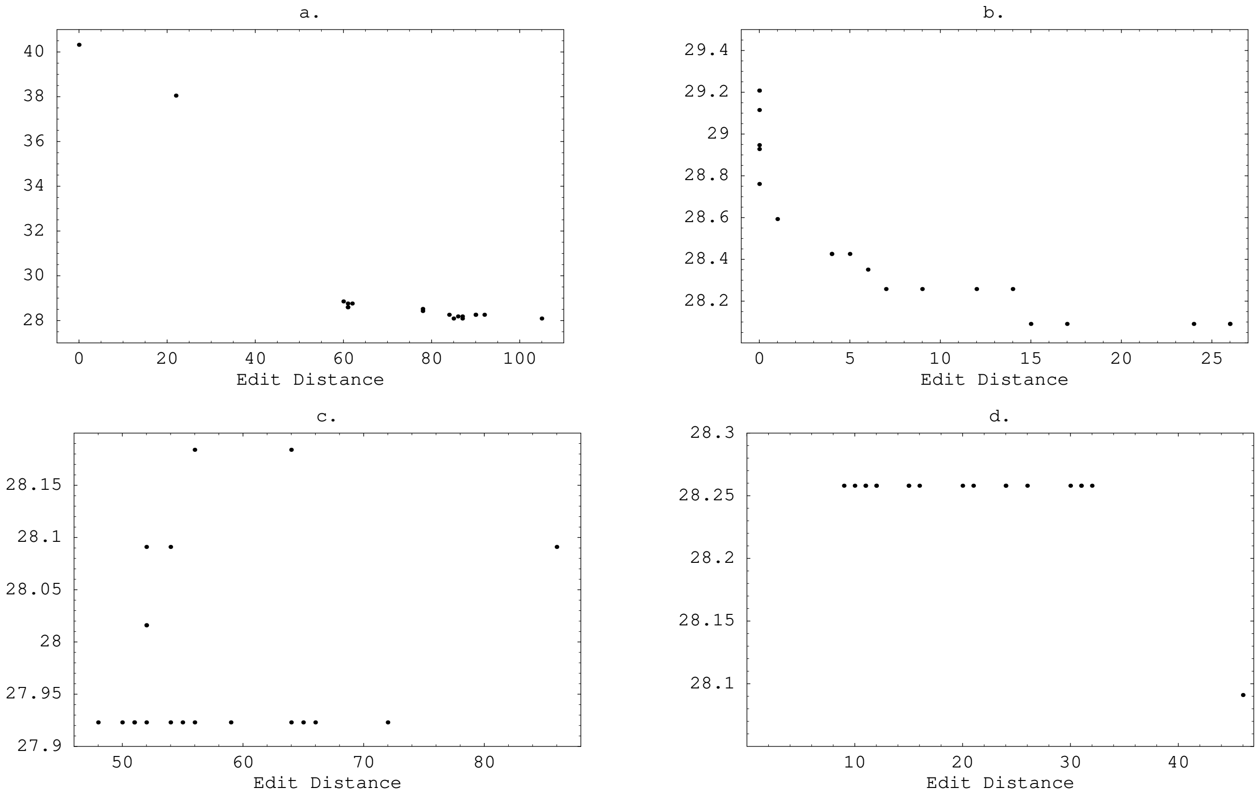

FIG. 5. *a. Unmodified edit distance in a ring topology passing single migrants. b. Modified edit distance, in which empty ply encodings are considered to match, in a ring topology passing single migrants. c. Modified edit distance and topology mapped communication grid, where each panel sends and receives data from each adjacent panel, passing single migrants. d. Modified edit distance using multiple migrants in the migrant box, ring topology.*

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. The scatter plots (Figure 5 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. 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 (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 perfectly blended, and it is lighter than the best (29.16 kg) fully blended design found by Sormekun et al. [11].

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

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	[		$\pm 30_2$	± 30	-75	75	-75	$\pm 60_3/75/15]_s$
2	2.51	[		± 30	± 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	[	± 30	$\pm 30_2$	± 30	75	60	-75	$\pm 60_3/75/15]_s$
10	3.02	[	30	$\pm 30_2$	± 30	75	60	-75	$\pm 60_3/75/15]_s$
11	2.51	[		± 30	± 30	75	60	-75	$\pm 60_3/75/15]_s$
12	2.51	[			± 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	$\pm 75_2$	± 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	$\pm 60_3/75/15]_s$

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 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.

5 Conclusions

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

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	$\mp 45/-75/\mp 60/60/45/75]_s$
2	2.51	$[$		30	± 30	-30	75	± 75	$\mp 45/-75/\mp 60/60/45/75]_s$
3	1.02	$[$					75	± 75	$\mp 45/-75/\mp 60/60/45/75]_s$
4	0.84	$[$						75	$\mp 45/-75/\mp 60/60/45/75]_s$
5	0.74	$[$							$\mp 45/-75/\mp 60/60/45/75]_s$
6	1.02	$[$					75	± 75	$\mp 45/-75/\mp 60/60/45/75]_s$
7	0.93	$[$						± 75	$\mp 45/-75/\mp 60/60/45/75]_s$
8	1.12	$[$				-75	75	± 75	$\mp 45/-75/\mp 60/60/45/75]_s$
9	3.35	$[$	$\pm 30_2$	± 30	± 30	-75	75	± 75	$\mp 45/-75/\mp 60/60/45/75]_s$
10	3.02	$[$	± 30	± 30	± 30	-75	75	± 75	$\mp 45/-75/\mp 60/60/45/75]_s$
11	2.68	$[$		± 30	± 30	-75	75	± 75	$\mp 45/-75/\mp 60/60/45/75]_s$
12	2.51	$[$		30	± 30	-75	75	± 75	$\mp 45/-75/\mp 60/60/45/75]_s$
13	1.02	$[$					75	± 75	$\mp 45/-75/\mp 60/60/45/75]_s$
14	0.84	$[$						75	$\mp 45/-75/\mp 60/60/45/75]_s$
15	1.12	$[$				-75	75	± 75	$\mp 45/-75/\mp 60/60/45/75]_s$
16	1.68	$[$	± 75	± 75	± 75	-75	75	± 75	$\mp 45/-75/\mp 60/60/45/75]_s$
17	0.93	$[$						± 75	$\mp 45/-75/\mp 60/60/45/75]_s$
18	1.02	$[$					75	± 75	$\mp 45/-75/\mp 60/60/45/75]_s$

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

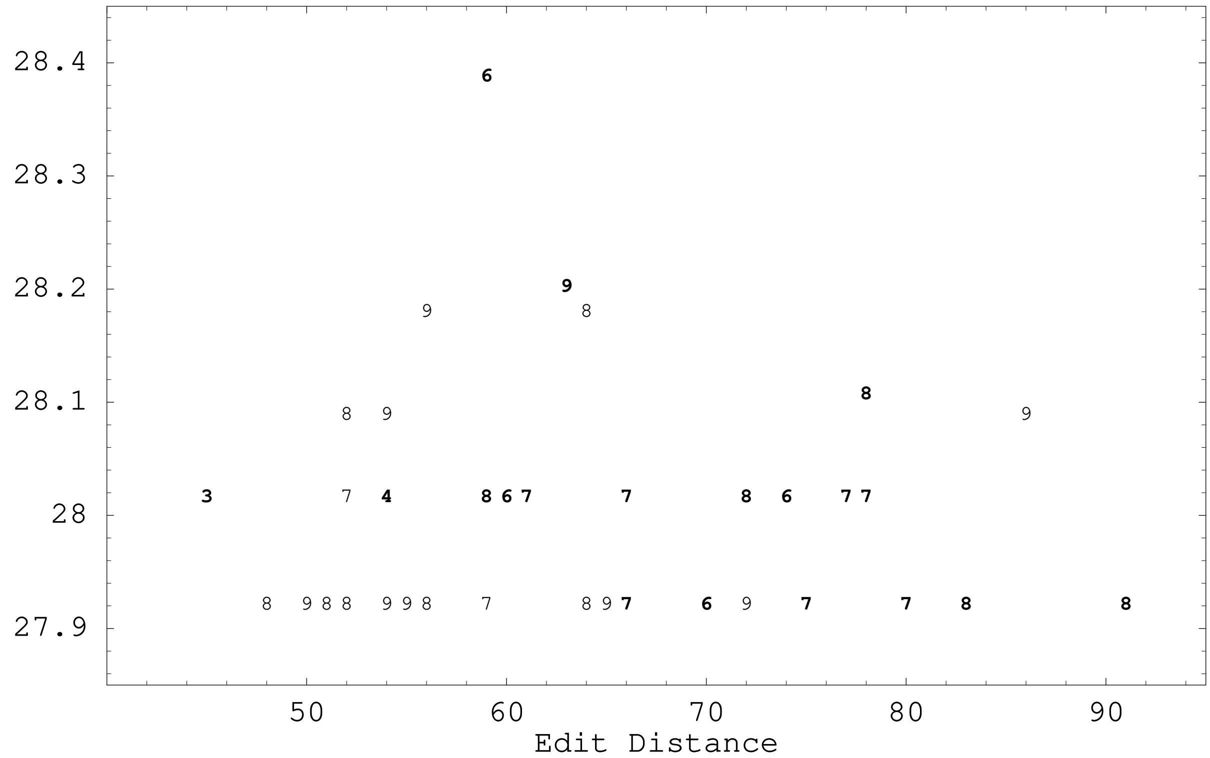

FIG. 6. *Each plotted point is rendered as an integer representing the maximum edit distance between any single pair of neighbors. Those points in non-bold font are the same points appearing in Figure 5c. They represent the topology mapped communication grid using modified edit distance and single migrant 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 Section 3.4.*

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.

References

[1] 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.

[2] 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.

[3] G. Soremekun, Z. Gürdal, R. T. Haftka, and L. T. Watson, *Improving genetic algorithm efficiency and reliability in the design and optimization of composite structures*, Technical Report TR-96-13, Dept. of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA, 1996.

[4] J. H. Holland, *Adaptation in natural and artificial systems*, The University of Michigan Press, Ann Arbor, MI, 1975

[5] 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.

[6] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*, Addison Wesley, Reading, MA, 1989

[7] T. Bäck, *Evolutionary algorithms in theory and practice*, Oxford University Press, NY, 1996

[8] 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.

[9] M. T. McMahon 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.

[10] 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.

[11] G. Sormekun, Z. Gürdal, C. Kassapoglou, and D. Toni, *Stacking sequence blending of multiple composite laminates using genetic algorithms*, in Proceedings of 42nd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, AIAA Paper 2001-1203, Seattle, WA, 2001.

[12] V. I. Levenstein, *Binary codes capable of correcting insertions and reversals*, Sov. Phys. Dokl., 10 (1966), pp. 707–710.

[13] D. Gusfield, *Algorithms on strings, trees, and sequences*, Cambridge University Press, New York, NY, 1997

[14] 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.

[15] 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.

[16] 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.