# A graph-based parameterization concept for global laminate optimization



**Originally published in:**
Structural and Multidisciplinary Optimization 36(3), https://doi.org/10.1007/s00158-007-0165-8

Struct Multidisc Optim (2008) 36:289–305
DOI 10.1007/s00158-007-0165-8

RESEARCH PAPER

# A graph-based parameterization concept for global laminate optimization

**M. Giger · D. Keller · P. Ermanni**

Received: 19 July 2006 / Revised: 18 May 2007 / Accepted: 28 May 2007 / Published online: 22 September 2007


**Abstract** A new graph-based parameterization concept aimed at the global optimization of laminated structures by the means of evolutionary algorithms and finite element analysis is introduced. The motivation to develop this novel parameterization concept is twofold. First, the entire design space is accessible to optimization down to the smallest entity, which is a single finite element, and secondly, this concept guarantees greatest flexibility in terms of laminate layer shape and placement. The finite element mesh of a structure is represented as a mathematical graph. Substructures of this graph form fiber reinforced and possibly overlapping patches and are affiliated to virtual graph vertices representing their properties. Adapted genetic variation operators are directly applied on this graph. The method allows for concurrent optimization of number, size, shape, and position of the patches and an arbitrary number of material related properties for each of them. The novel concept overcomes the limits of traditional geometry-based approaches, as it is able to represent almost arbitrary patch shapes even on curved surfaces. Two numerical examples demonstrate the efficiency of the method.

M. Giger (✉) · D. Keller · P. Ermanni
ETH Zurich, Centre of Structure Technologies,
Leonhardstrasse 27, CH-8092 Zurich, Switzerland
e-mail: mathias.giger@gmail.com

D. Keller
e-mail: davidkeller@ethz.ch

P. Ermanni
e-mail: permanni@ethz.ch

**Keywords** Laminate optimization · Mathematical graph · Structural optimization · Parameterization · Evolutionary algorithms

## 1 Introduction

Today laminated composites are very popular in aerospace, marine, automotive, and sport applications due to their excellent mechanical properties as well as the adjustability of these through the combination of different laminate. However, an optimal layout of a composite structure is still hard to find: complex geometries, different materials, and manufacturing as well as economical limitations typically lead to highly constrained problems with many parameters.

A lot of research has been done in the field of composite optimization; for a thorough review, see Venkataraman and Haftka (1999) and many references therein. Furthermore, many publications treat the optimization of composite plates with holes for increased strength (Huang and Haftka 2005, and references therein). The optimal orientation of orthotropic materials is investigated by Pedersen and others (Pedersen 1989, 1991, 2004; Hammer et al. 1997).

The parameterization is one of the most important issues in computational structural optimization, as it may strongly influence the quality of the optimized structure. There are a few different approaches for the parameterization of laminate optimization problems. The most basic one is the representation of the stacking sequence by a material and a ply angle parameter for each layer (e.g., Le Riche et al. 1995 and Grosset et al. 2001). A further refinement considers the number of

layers as an optimization parameter that leads to a variable number of parameters for different design solutions. Apparently, this parameterization suffers from its global point of view in case a single stacking sequence holds for the entire laminate, i.e., the entire mechanical structure. Most often, structural failure and constraint violations are local effects; hence, an optimal laminate representation should allow local reinforcements. A representation that divides the design space into different regions (cells) with their individual laminate properties overcomes this disadvantage. The laminates of adjacent cells should have some common (i.e., global) layers to ensure the cohesion of the whole structure. Thus, the problem is to encode the stacking sequence and the shape of these regions without introducing an unnecessary high number of additional constraints.

The *patch concept* proposed by Zehnder and Ermanni (2006) convinces with its manufacturing-oriented approach. One *global* layer is called a patch, and a variable number of these patches are arranged in the design space finally building the laminates of the entire mechanical structure. Such layers are called global, as they can be positioned anywhere in the design space without any limitations except the design space boundaries. Their positions, sizes, shapes, ply angles, and materials are changed during optimization, whereas the order of the patches in the parameter set (i.e., genotype) represents the stacking sequence of the laminate in the analysis model. However, the shape of the patches is somehow limited by the computer-aided design-based encoding as parameterized sketches. Prior knowledge is required to create associative patch prototypes without unnecessarily narrowing the optimization possibilities.

The inhomogeneous, variable length representation claims certain capabilities of an optimization algorithm:

*Material decision variables* Patch materials are typically to be chosen out of an existing list. This list has neither natural order nor norm. A material comes along with a set of coupled parameters like Young's modulus, Poisson's ratio, thickness, price, etc.

*Continuous decision variables* Geometry and possibly other material related parameters can be continuous.

*Discrete decision variables* Manufacturing accuracy possibly requires the ply angles to be discrete.

*Multi-objective or multimodal* Economical, manufacturing, and engineering constraints lead to multimodal and multi-objective problems, which can be transformed to a single multimodal objective through introduction of penalty terms. Therefore, a global optimization strategy is required.

Evolutionary algorithms are able to meet all these requirements (e.g., Michalewicz and Schoenauer 1996 or Michalewicz et al. 1996). Two-dimensional topology optimization problems are somehow similar to laminate optimization, and evolutionary approaches prosper there as well, e.g., Kane and Schoenauer (1996).

Giger and Ermanni (2006) already use a graph-based genotype for the representation of truss structures. This work presents a similar graph-based patch parameterization concept for laminate optimization representing the finite element mesh with a mathematical graph abstraction and allows for almost arbitrary patch geometries. A single patch is represented by a subgraph including only a subset of all finite elements, whereas such a subgraph is connected to an additional virtual vertex holding the properties (e.g., material, ply angle, etc.) of the respective patch. As the optimization is done with an evolutionary algorithm, a set of new genetic variation operators is introduced to customize the algorithm for the graph-based genotype.

First, the evolutionary algorithm used in this work is presented in Section 2, and a brief introduction to the basic graph terminology is given in Section 3. Then, the graph-based parameterization concept is explained in Section 4. Section 5 gives an overview over the evolutionary operators particularly adapted to the presented genotype, and Section 6 shortly discusses some implementation details. Finally, a rather academic as well as an engineering problem illustrate the performance and capabilities of the new method in Section 7 before the paper is concluded with Section 8.

## 2 Evolutionary algorithm

The graph-based parameterization concept is developed for the paradigm presented in Fig. 1.

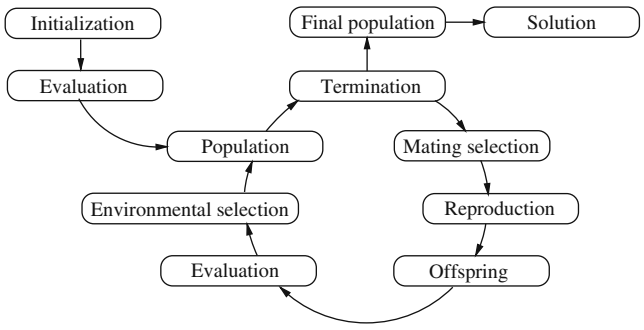
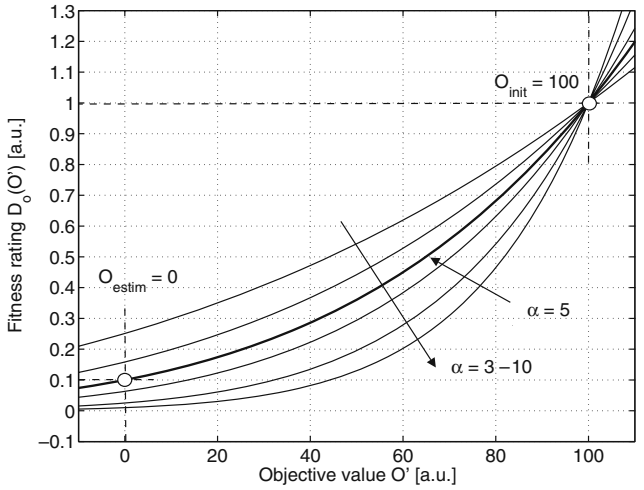



**Fig. 1** Evolutionary algorithm scheme

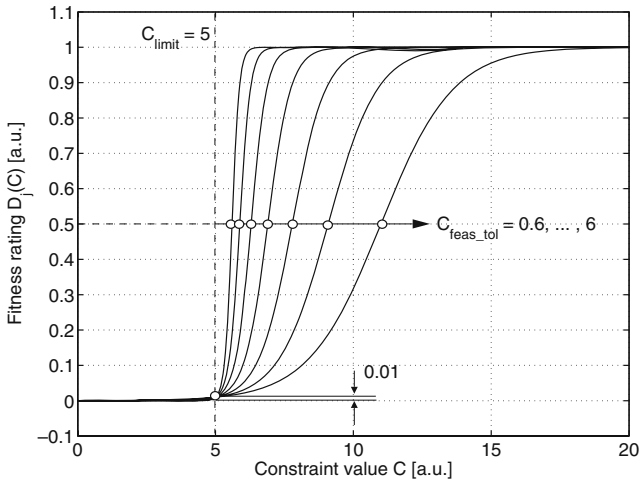
After a first population of a predetermined size $P$ is randomly initialized, all individuals are evaluated, and a fitness value is assigned to each of them. The fitness value is a weighted sum of the design objective and the given limit constraints as shown in the following equation

$$F(\mathbf{i}) = w_o D_o(\mathbf{i}) + \sum_j w_j D_j(\mathbf{i}). \quad (1)$$

Example fitness definition functions are shown in Fig. 2, whereas the objective function is defined by the parameters $O_{\text{init}}$, $O_{\text{estim}}$, and $\alpha$ and the constraint functions are defined by $C_{\text{limit}}$ and $C_{\text{feas\_tol}}$. Depending on the optimization, these fitness functions have to be adjusted to guide the optimization process toward superior solutions. The complete mathematical definitions of these functions can be found in König (2004).




**a** Design objective




**b** Limit constraint

**Fig. 2** Example fitness definition functions

With the initial population, the optimization loop is entered. In the mating selection stage, the individuals that are allowed to reproduce themselves are chosen by a deterministic tournament selection procedure. In this work, a tournament selection size of 2 is used. The tailored genetic operators, i.e., crossover and mutation operators presented in Section 5, recombine the genetic material of the chosen individuals and introduce new genetic material by mutating random parts of the genotypes in the reproduction stage. The resulting offspring solutions are then evaluated, and again, fitness values are assigned. In the environmental selection stage, it is determined which offspring solutions replace individuals of the current population. For the optimization problems presented in this work, a generational approach is used, which replaces all the parents by offspring solutions; only the overall best solution is always kept in the population; that is, a "weak elitism" mechanism is employed. This optimization loop is run until a given termination criterion is met. Typically, the optimization is stopped after a given number of generations or evaluations.

## 3 Basic graph terminology

Mathematical graph theory is one of the most active and complex fields of research. Therefore, it is preferable to introduce only the terminology and the basic concepts used within this work instead of presenting an extensive overview at this point.

A graph is an abstract mathematical model: It is a pair $(V, E)$, where $V$ is a finite set called vertex set and $E$ is a binary relation on $V$ called edge set. Elements of $V$ are called *vertices*, elements of $E$ *edges*. An edge is a pair $(u, v)$ with $u, v \in V$ (Siek et al. 2002).

In *undirected* graphs edges are unordered pairs, therefore $(u, v)$ and $(v, u)$ represent the same edge. As further restrictions self-loops, i.e., an edge with identical source and target vertices $(u, u)$, and parallel edges connecting the same vertices are not allowed for the graphs used in this work. The *degree* of a vertex is the number of edges incident to it. In a *regular* graph of degree $r$, every vertex must have a vertex degree of $r$. Edges as well as vertices can take representation-dependent properties, which allow to adapt and extend graph concepts on many different problems. Typical applications of graph theory can be found in electric engineering, network and traffic management as well as algorithms and data structures for computation.

## 4 The graph-based parameterization concept

A commonly used method for the numerical simulation of the mechanical behavior of a mechanical structure is the finite element method (FEM; e.g., Zienkiewicz and Taylor 2000). FEM requires a discretization of the mechanical structure into small (i.e., finite) elements. This so-called *mesh* consists of elements and their associated nodes. Its structure, e.g., the number of nodes per element or the shape of the elements, depends on the element type(s) used.

### 4.1 Graph abstraction of finite element meshes

The information embodied in such a finite element mesh can be transferred into different graph abstractions as it is illustrated on a basic two-dimensional four-element mesh in Fig. 3. A *node graph* (Fig. 3a) represents the contiguity information of the mesh nodes (a vertex represents a mesh node, and neighboring nodes are connected with an edge). It may be used for node reordering to minimize the bandwidth of the stiffness matrix of the numerical model. For this purpose, information on the exact geometric position of the nodes is not required and would therefore not be part of the graph abstraction. The second graph (Fig. 3b) represents the finite elements as additional vertices of a special quality and introduces an adjacency information between elements and their nodes. This model could represent a data structure of a finite element implementation. The *element graph* (Fig. 3c) contains the element adjacency information of the mesh. The laminate optimization method presented in this work operates on this information. Three definitions are given to clarify the concept of the element graph.

**Definition 1** (Element vertex) An element vertex is an abstraction of one finite element. It stores the nodes and the properties of the underlying finite element.

**Definition 2** (Element graph) The element graph of a finite element mesh contains element vertices of all finite elements. Two finite elements with *more* than one coincident mesh node are connected with an edge.

**Definition 3** (Adjacency set) The adjacency set of one element vertex consists of its adjacent element vertices. The adjacency set of an element vertex is named *full* if there are as many adjacent element vertices as sides in the respective element.

These very general definitions theoretically hold for any kind of finite element model. In this paper, only the special case of laminated structures is considered. Such composite structures are usually thin walled, three dimensional, and in laminar shapes that can be modelled with shell elements. Special types of these finite

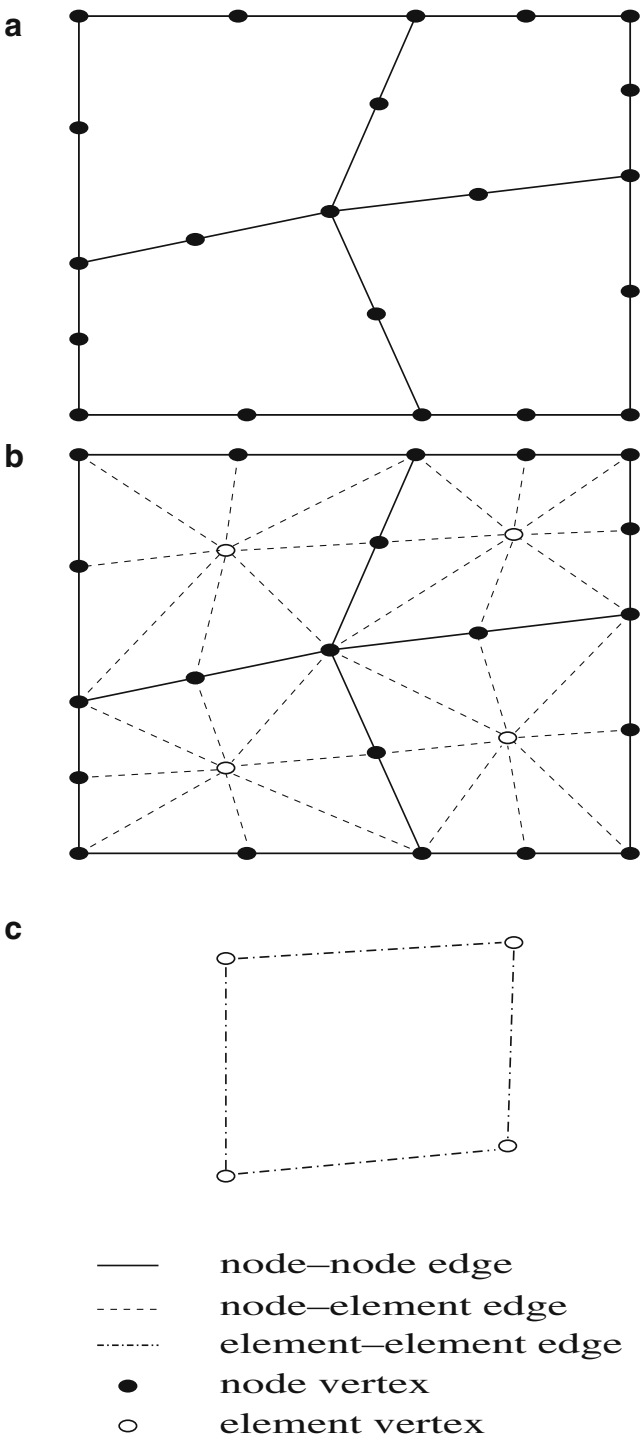



**Fig. 3** Graph abstraction of two-dimensional finite element meshes: **a** direct interpretation of the finite element mesh as a graph based on its nodes (*node graph*). **b** Additional virtual element vertices. **c** Connection of the virtual element vertices (*element graph*)

elements are able to model a layered laminate assembly and are commonly used in a rectangular and a triangular implementation with eight or six nodes, respectively. In general, vertices in the element graph have at most four adjacent vertices in their rectangular representation and three for the triangular version. The element graph of a laminated structure is typically not regular. Not only vertices on the boundaries of the structure have a lower degree, but often, there are differences in degree due to the different element implementations used to mesh a geometry.

The graph-based laminate optimization method is limited to surfaces consisting of simply connected areas (e.g., composite panels). More complex structures, e.g., beams with T-cross-sections, cannot be entirely optimized yet, because two or more surfaces are connected to each other along a common line. Depending on the normal direction along which the stacking sequence is evaluated, this could lead to intersecting layers that is impossible from a manufacturing point of view. However, such adjacent surfaces could be optimized independently.

## 4.2 The graph-based laminate genotype

In evolutionary computation, the genotype represents the information on which evolution acts. Before evaluation, it is mapped to the phenotype space, where environmental pressure occurs. Therefore, the possibilities, the efficiency, and also the compliance with constraints of evolutionary computational optimization highly depend on the characteristics of the genotype. First, the parameterization of a single patch is presented, and the parameterization of the entire laminate follows afterward.

### *4.2.1 Patch parameterization*

**Definition 4** (Patch) According to Zehnder and Ermanni (2006), a patch is one global layer (one constitutive component) of a laminated structure.

The following requirements should be fulfilled by the genotype representation of one patch:

**Discretization** In the phenotype space, one finite element is the smallest inseparable building block of a patch. This is a result of the discretization with layered elements during evaluation. Once a finite element mesh is regarded as fixed, a genotype that offers higher resolution than this mesh sacrifices efficiency. A flexible mesh demands remeshing of the structure before every evaluation, which increases the computational costs for each evaluation. This work follows a fixed-mesh approach. Therefore, a set of dedicated finite elements incorporates the shape, the size, and the position of one patch.

**Properties** One patch can take several properties that can be considered as coupled or independent optimization parameters: e.g., material, ply angle, thickness, etc.

**Connectivity** A patch is a connected region on the shape of a mechanical structure. Its dimensions are limited by the boundary of the mechanical structure and possibly by further constraints like a maximum number of finite elements belonging to the respective patch.

These requirements can be satisfied with the following approach: The genotype is based on the element graph of the mechanical structure. This graph stays the same during the optimization; that is, no remeshing takes place. For every patch, a virtual *patch vertex* is added to the graph.

**Definition 5** (Patch vertex) A patch vertex is an abstract entity representing the properties of one patch.

Thus, a patch vertex has special qualities. It takes a heterogeneous set of representation-dependent genes to encode the properties of its patch (e.g., material, ply angle, etc.). The term gene in this context is used in the sense of a single-valued parameter after the idea of the *universal gene* introduced by König (2004).

Each finite element within the shape of one patch is further connected by an edge with the newly introduced patch vertex (Fig. 4). The connectivity of the patch shape is not guaranteed; it has to be achieved with functional refinements.

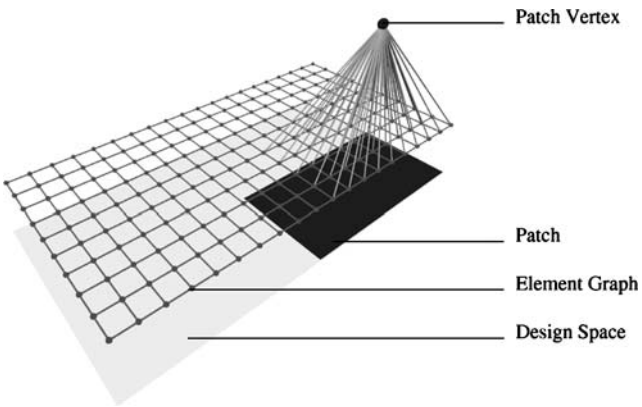



**Fig. 4** Graph-based representation of one patch

**Definition 6** (Patch graph) A subgraph including all adjacent vertices of one patch vertex and their connecting edges is called a patch graph.

**Definition 7** (Patch gene) The information represented by a patch vertex in conjunction with its affiliated genes and its adjacent edges encodes one physical patch and is further on called patch gene.

Generally, the laminate structure consists of a variable number of patches, whereas the stacking sequence of the laminate at a given point still needs to be defined.

*4.2.2 Laminate parameterization*

A laminated structure is represented as a set of patch genes organized in a genotype vector. The genotype vector shown in Fig. 5 consists of three patch vertices, whereas each of these patch vertices belongs to exactly one patch gene defining one physical patch. The order of the patch vertices in the variable-length genotype vector, i.e., the number of patches can be varied, encodes the stacking sequence of the laminate. As the finite element mesh consists of layered shell elements, the properties of the patches, namely material and orientation, can be mapped to the finite elements that are covered by the respective patches. Consequently, the connectivity of different layers is ensured even if a finite element is covered by only two layers that are not adjacent in the genotype vector. Such a genotype contains information on three levels:

**Laminate level** describes the set and the sequence of the patches analogous to a global lamination plan. It is represented by the variable-length genotype vector.

**Patch level** encodes the size, the shape, and the position of one patch. One patch gene incorporates this information.

**Patch property level** contains properties of one patch. They are encoded as genes in a patch vertex.

4.3 Patch structure manipulation

The structure of a patch needs to be changed to create new design solutions during the optimization process. All structural patch manipulations can be interpreted as either removing or adding edges between a patch vertex and some element vertices in the element graph. Each modification changes the patch graph defining one patch. The graph abstraction of the patch allows to efficiently analyse the patch graph for connectivity, size, shape, and position.

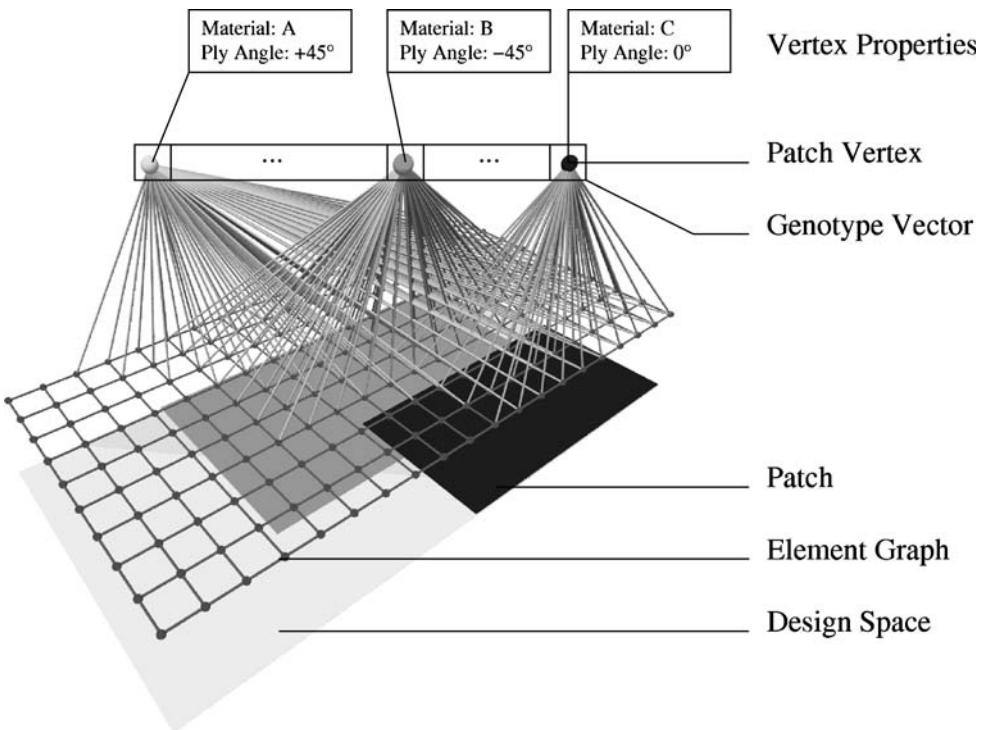



**Fig. 5** Graph-based laminate representation. The depicted laminate consists of three patches, and the stacking sequence is [45°/−45°/0°] (the genotype vector is processed from left to right) where all three patches overlap each other, but most of the finite elements are covered by less patches; hence, their stacking sequences are accordingly reduced

#### 4.3.1 Regions of a patch

Adjacency information allows to specify different regions of a patch as shown in Fig. 6:

*Patch elements* are all finite elements in the patch graph.

*Patch core elements* are vertices in the patch graph with a full adjacency set.

*Boundary elements* are patch members with a reduced set of adjacent patch elements: *Design space boundary elements* have no adjacent elements outside of the patch, *cambium elements* have adjacent elements outside of the patch and, *articulation elements* connect two regions of a patch graph. Removing one articulation element splits the patch graph into different components.

*Neighbor elements* are not patch elements but are adjacent to at least one patch element.

#### 4.3.2 Basic growth and shrinking mechanisms

Growth and shrinking operations are elemental patch variations. They are base components of the genetic operators presented in Section 5, and their application in stochastic search algorithms demands a random behavior.

Engineering as well as manufacturing require a patch shape to stay *compact* and *connected*. To ensure this, growth mechanisms are limited to add neighbor elements, and shrinking mechanisms only remove boundary elements (articulation elements are handled in a special way).

The *neighborhood size concept* is introduced to describe the environment of element vertices.

**Definition 8** (Neighborhood size) The neighborhood size $\eta(v_i, \pi)$ of one element vertex $v_i$ in respect of the patch $\pi$ is the number of its adjacent patch elements.

**Definition 9** (Relative neighborhood size) The relative neighborhood size $\rho(v_i, \pi)$ of one element vertex $v_i$ in respect of the patch $\pi$ is its neighborhood size $\eta(v_i, \pi)$ divided by the number of possible adjacent elements $p$.

The number $p$ only depends on the element type of $v_i$; it equals four for rectangular elements and three for triangular elements.

As a selection criterion for elements to be removed or added to a patch, the neighborhood size concept promotes compact patch shapes. Growth and

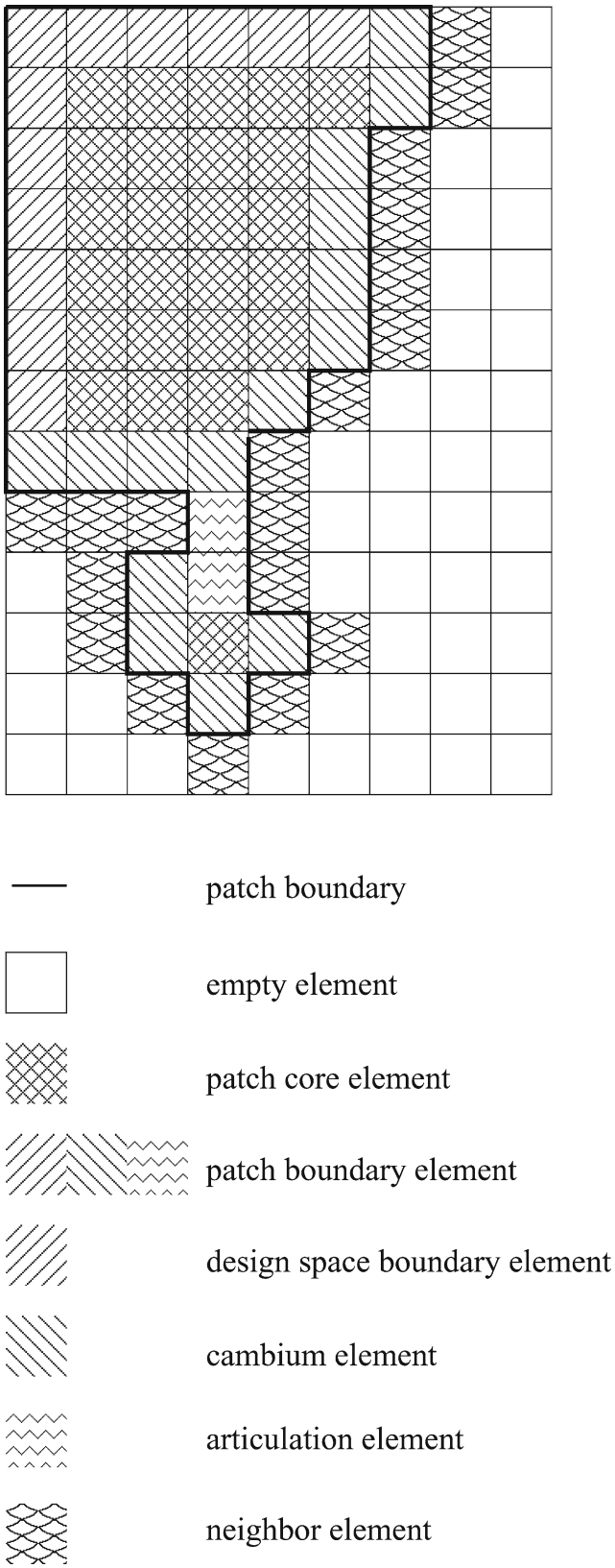



**Fig. 6** Illustration of different patch regions on a regular, rectangular finite element mesh. "Empty elements" are finite elements that are not covered by the patch

shrinking mechanisms use a random weighted selection procedure, where the weight factors depend on the relative neighborhood size of the respective elements.

**Parallel growth mechanism** The parallel growth mechanism resizes a patch by adding a certain predefined number of edges between a patch's neighbor elements and its patch vertex. The method performs a random weighted selection out of all neighbor elements, where the weight for each neighbor element is equal to its relative neighborhood size.

**Parallel shrinking mechanism** The parallel shrinking mechanism works analogously to the parallel growth mechanism. It removes edges between a patch's boundary elements and its patch vertex. There are two variations of the method: The first one randomly selects out of a pool of all boundary elements; the second one's pool contains all boundary elements except articulation elements. The selection weight is one minus the relative neighborhood size of the respective element. The first variation does not protect the connectivity of the patch shape.

**Recursive growth mechanism** While the parallel mechanisms perform variations on the whole shape of a patch, the recursive methods work more locally:

1. Initialize a recursion counter $i = 0$.
2. Choose a random weighted cambium element $v$ (the weight is the relative neighborhood size of the respective element).
3. Adjacency iteration:
   (a) For each adjacent neighbor vertex $v_a$ of $v$ make a random choice with probability $P_g(i, v_a)$ for *true* out of [*true*, *false*].
   (b) If the result is *true*, add an edge between the patch vertex and $v_a$ and start a new instance of the process with $v = v_a$ and $i = i + 1$ at step 3a.
   In any case (true or false), continue the current instance at step 3c.
   (c) Continue with the next adjacent neighbor vertex $v_a$ of $v$ at step 3a.

The probability $P_g(i, v_a)$ is defined as:

$$P_g(i, v_a) := e^{-\frac{i}{d_g \cdot \rho(v_a, \pi)}}, \tag{2}$$

where $d_g$ is a user-defined growth parameter. There is no active termination criterion. Therefore, a patch can theoretically grow until it reaches the boundary of the design space. However, this is not very probable, as $P_g$ diminishes fast with increasing recursion depth $i$. In regular unlimited graphs, this method creates semicircular excrescences.

**Recursive shrinking mechanism** The recursive shrinking mechanism works analogously to the recursive growth method:

1. Initialize a recursion counter $i = 0$.
2. Choose a random weighted boundary element $v$ (the weight is the relative neighborhood size of the respective element).
3. Edge removal:
   (a) Remove the edge between the patch vertex and $v$.
   (b) If the patch size reaches a user-defined minimal size stop the process.
4. Adjacency iteration:
   (a) For each adjacent patch, vertex $v_a$ of $v$ makes a random choice with probability $P_s(i, v_a)$ for *true* out of [*true*, *false*].
   (b) If the result is *true*, start a new instance of the process with $v = v_a$ and $i = i + 1$ at step 3a.
   In any case (true or false), continue the current instance at step 4c.
   (c) Continue with the next adjacent vertex $v_a$ of $v$ at step 4a.

The probability $P_s(i, v_a)$ is defined as:

$$P_s(i, v_a) := e^{-\frac{i}{d_s \cdot (1 - \rho(v_a, \pi))}}, \tag{3}$$

where $d_s$ is a user-defined shrinking parameter. A variation of the method ensures the connectivity of the patch graph by excluding articulation elements out of the selection pool.

## 5 Genetic graph operators

In evolutionary computation, it is common practice to divide variation operators into mutation and crossover operators. Mutation operators are unary operators; that is, they typically perform slight changes on the genetic information of a single individual. Crossover operators act on two or more individuals by exchanging genetic information between them. As the genotype groups genetic information on different levels, namely laminate, patch, and patch property level (see Section 4.2.2), there are different types of operators for each level.

## 5.1 Mutation operators

### *5.1.1 Laminate mutation operators*

**Remove and add patch mutation** either remove a randomly chosen patch gene or insert a new randomly initialized patch gene at a random position in the genotype.

**Swap patch mutation** swaps two patch genes in the genotype. This changes the stacking sequence of the laminate.

### *5.1.2 Patch mutation operators*

**Parallel resize mutation** resizes a patch to a randomly chosen size by either applying parallel growth or parallel shrinking.

**Recursive resize mutation** resizes a patch by recursive growth and shrinking mechanisms, whereas the final number of elements in the patch results from the random termination of the resizing mechanism.

**Random walk mutation** removes one random boundary element $v_b$ from a patch and adds a random neighbor element $v_n$ that is close to $v_b$. $v_b$ is randomly chosen out of all boundary elements (uniform probability); $v_n$ is the exit point of a random walk through the patch graph.

**Bobtail mutation** renders a patch shape more compact by removing one randomly chosen articulation element and the smaller of the remaining components from a patch.

### *5.1.3 Property mutation operators*

**Uniform mutation and Gauss mutation** operate on patch vertex properties. Typically, these patch vertex properties are encoded with universal genes (König 2004) providing these standard mutation operators.

## 5.2 Crossover operators

### *5.2.1 Laminate crossover operators*

**Uniform crossover** takes two parent individuals and swaps two randomly chosen patch genes a predefined number of times.

**One point crossover** takes two parent individuals and selects one random crossover point for each. The head of the first individual is recombined with the tail of the second one and vice versa.

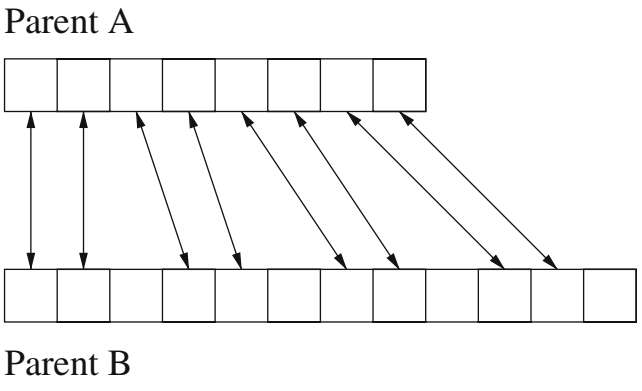



**Fig. 7** Possible patch gene exchanges in intermediate crossover

**Two point crossover** takes two parent individuals and selects two random crossover points for each. The segments of each individual between the crossover points are exchanged.

**Intermediate crossover** takes two parent individuals and randomly exchanges patch genes with a given probability, but keeps the stacking sequence (Fig. 7).

### *5.2.2 Patch crossover operators*

**Marry crossover** takes two parent individuals and creates one offspring that is based on the first parent. It chooses two patch genes out of its parents at random and searches for the shortest path between the two patch vertices of the patch genes. Each element of the second parent's gene and on the connecting path are added to the first parent. In a second step, a recursive growth is started on each element on the path to create a more compact shape, whereas the growth rate is a user-defined parameter. The modified first parent becomes the offspring individual.

**Position crossover** takes two parent individuals and creates one offspring that is based on the first parent. It searches again for the shortest path from one randomly chosen patch gene from the first parent to another one from the second parent. From the middle of this path, a new patch grows (recursive and parallel growth) to a random size in between the size of the two parent patches. This new patch replaces the original patch from the first parent, which then becomes the offspring individual.

**Overlap crossover** selects three parent individuals and creates three offspring (Fig. 8). The parent selection searches for three parent individuals having overlapping patches (A, B, C), whereas patch A belongs to the first, patch B to the second, and patch C to the third

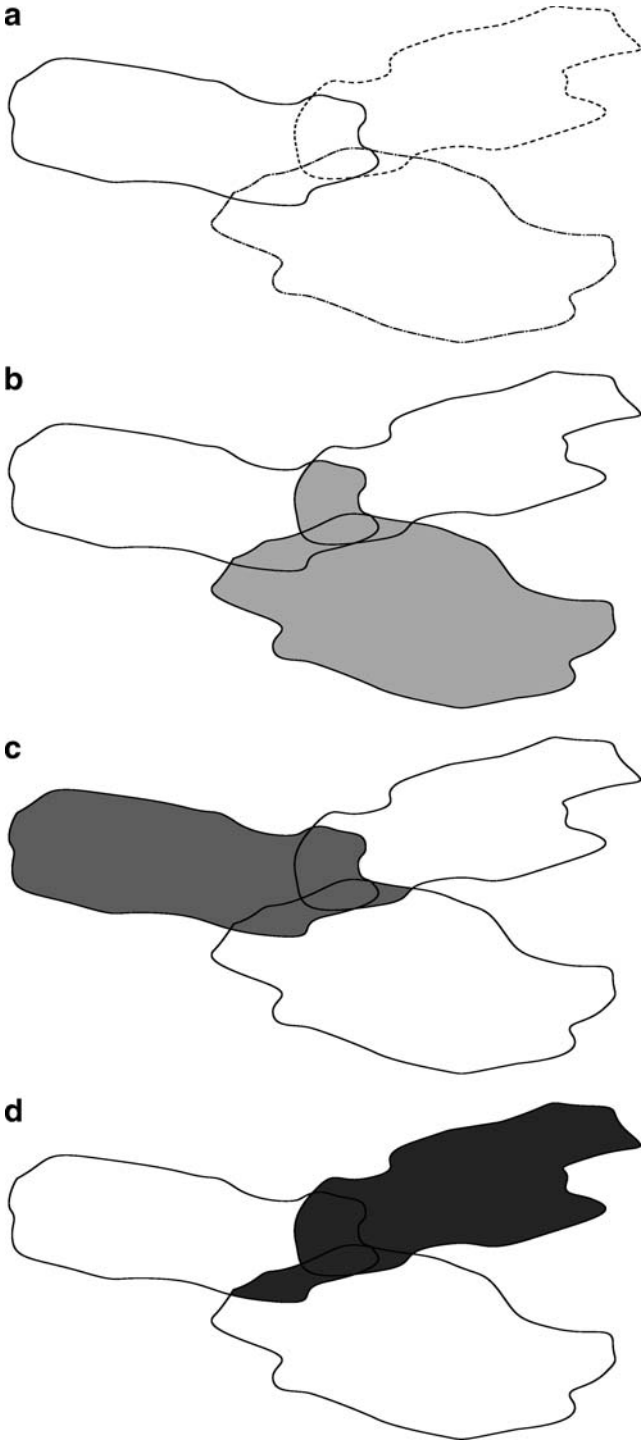



**Fig. 8** Overlap crossover: **a** Three overlapping parent patches **b** offspring 1 **c** offspring 2 **d** offspring 3

parent, respectively. The overlapping parts of patch B and C are added to patch A, those of A and C to patch B, and those of A and B to patch C. Afterward, the modified parents become the offspring. This way, the operator exchanges geometric subsets between patches even on curved shapes and in irregular element graphs.

*5.2.3 Property crossover operators*

**Segment crossover** is an arithmetical crossover determining new gene values $a$ and $b$ according to the following rule:

$$a = A \cdot \gamma + B \cdot (1 - \gamma) \tag{4}$$

$$b = A \cdot (1 - \gamma) + B \cdot \gamma, \tag{5}$$

where $A$ and $B$ are the parent individual values and $\gamma$ is a random proportional factor $\in [0, 1]$. This operator is defined for each universal gene type (König 2004), i.e., integer genes, double genes, etc.

## 6 Implementation

The method is implemented in C++. The evolution engine is taken from the Evolving Objects[1] library. This library provides generic functional objects required for evolutionary computation and allows for optimization of almost any type of genotype. The graph data structures and graph algorithms derive from the Boost Graph Library[2]. The patch property genes are a part from the *eoUniGene* library introduced by König (2004). Finite element analysis is performed with FELyX,[3] an open source finite element library that is entirely written in C++. Therefore, it allows a direct integration into the optimization environment that speeds up mapping and evaluation operations.

## 7 Applications

7.1 Eigenfrequency optimization of a rectangular plate

*7.1.1 Problem*

This first application deals with a rectangular plate that is clamped at all edges; that is, all boundary nodes cannot move in $x$-, $y$-, and $z$-directions, but the rotations are free. A mass is placed on it according to Fig. 9, and the objective is to maximize the first eigenfrequency $f_1$ by placing some fiber-reinforced patches on the plate. All patches consist of the same unidirectional material (UD) as specified in Table 1, and elements that are not covered by patches take a very compliant fill-material. Two variations of the problem are analyzed. First, only a single patch is placed on the rectangular

[1] http://eodev.sourceforge.net
[2] http://www.boost.org
[3] http://felyx.sourceforge.net

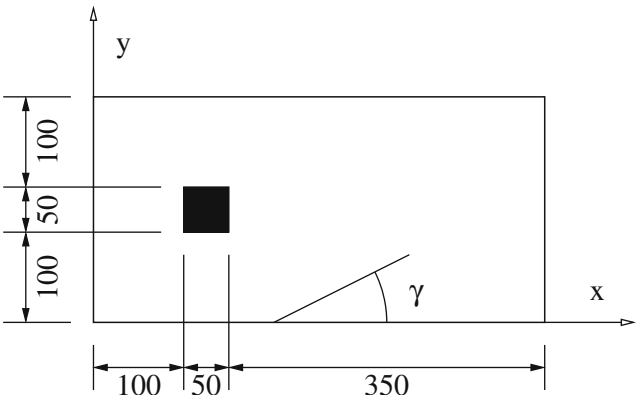



**Fig. 9** Rectangular plate with mass (*dark*). $\gamma$ is a possible fiber orientation angle relative to the global $x$-direction. All dimensions are in millimeters

plate to analyze the behavior of the applied genetic operators (except the overlap crossover). Afterward, this academic example is investigated with a maximum number of 12 patches. For the first example, the global mutation rate is set to 0.45, and the global crossover rate is equal to 0.48. For the second example with 12 patches, the global mutation rate is slightly reduced to 0.4, and the global crossover rate is decreased to 0.45. For both examples, the relative operator rates for all the different mutation and crossover operators presented in Section 5 are uniform.

### *7.1.2 Single patch optimization*

In the first verification setup, the number of patches is limited to one with a maximum size of 40% of the rectangle area. The ply angle is represented as a discrete optimization parameter that allows for values in the range from $-90°$ to $+90°$ in steps of $5°$. The fitness definition is taken from König (2004) mapping the objective, i.e., the first eigenfrequency, to a design objective value in [0, 1]. Furthermore, the maximum size of the patch is controlled with a limit constraint value in [0, 1] keeping the maximum size below 40% of the rectangle area. The overall fitness of an individual is then calculated as the sum of the design objective and the limit constraint value. Two different discretizations, with 200 and 800 finite elements, are optimized in populations of 100 individuals over 1,500 generations. The optimization formulation can therefore be stated as follows

$$\begin{aligned} &\text{maximize} && f_1 \\ &\text{subject to} && n_{pe200} \leq 80 \\ & && n_{pe800} \leq 320, \end{aligned} \tag{6}$$

where $n_{pe200}$ and $n_{pe800}$ denote the number of finite elements covered by the single patch.

**Table 1** Materials of the rectangular plate optimization

| Material | UD | Fill-material | Mass | |
|---|---|---|---|---|
| $E_1$ | 135 | 1 | 1 | [GPa] |
| $E_2$ | 10 | 1 | 1 | [GPa] |
| $\nu$ | 0.3 | 0.3 | 0.3 | [-] |
| $G_{12}$ | 5 | 0.5 | 0.5 | [GPa] |
| Thickness | 0.00015 | 0.00015 | 0.01 | [m] |
| Area density | 0.237 | 0.237 | 158 | [kg/m$^2$] |

**Results** Figure 11 depicts the convergence plots averaged over 12 runs for both variations, whereas all the runs reliably converge to similar final solutions. The solution with the highest eigenfrequency found for the 200 element representation has a first eigenfrequency of 1.36112 Hz (1.35624 Hz when evaluated with the 800 element finite element mesh); the one with 800 elements has a first eigenfrequency of 1.36387 Hz. The shapes of the best patches are shown in Fig. 10, and they have a fiber orientation of 90°. The 200 element discretization has the maximum allowed size of 40% of the rectangle area, whereas the 800 element discretization could have four more finite elements. If the optimization process is continued for the 800 element model, it

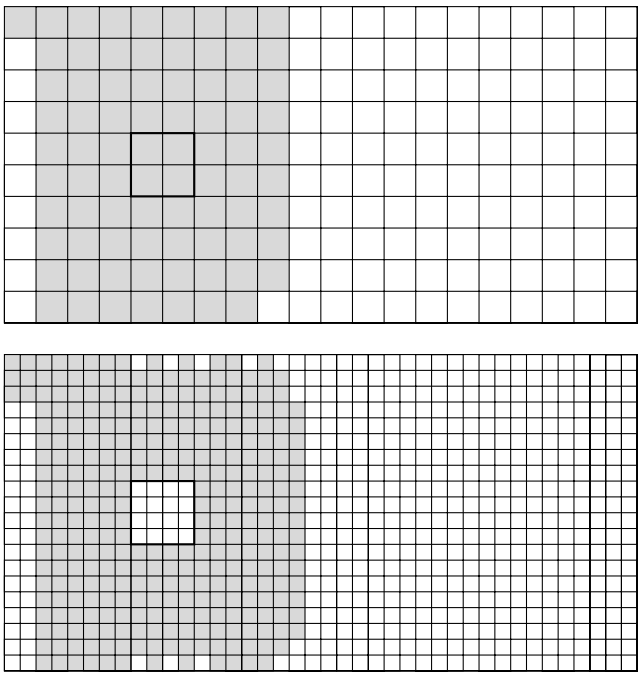


**Fig. 10** Solutions with the highest first eigenfrequencies complying with the size constraint after 1,500 generations with population size 100 of the single patch rectangle problem for 200 and 800 finite elements. The fiber orientation is 90° for both solutions

could be expected that additional four finite elements would also be covered contributing to an increased first eigenfrequency.

The method finds a partially similar solution for both discretizations. The region where the stiff material is placed, the size as well as the ply angle is about what one would expect as an optimal layout. Both solutions show compact patch shapes without any thin branches, and they almost perfectly reflect the $x$-symmetry (Fig. 9) of the problem. In the case of the 200 element discretization, a further improvement can hardly be found. If the top left finite element would be moved to the lower right (empty) corner of the patch, this would lead to the same eigenfrequency up to the fifth decimal place. Surprisingly, the 800 element solution shows a hole in the region of the mass that profits from the stiffness of the mass, see Table 1. There is actually no variation operator that actively creates a hole in a patch shape; hence, it is the result of a chain of different variations. The global optimum solution cannot be found for this example due to the relatively large number of degrees of freedom and the applied genetic operators incapable of slightly changing the boundary regions of the patch. Nevertheless, this example shows the capability of the patch concept to locally adapt to the higher stiffness of the mass region by leaving these finite elements blank. The convergence plots (Fig. 11) show that this mesh-based approach looses efficiency with an increasing number of elements. However, the 200 element discretization already finds fairly well solutions after 200 evaluated generations, and the 800 element discretization finds near-optimum solutions after 500 generations.

### *7.1.3 Constrained optimization*

In a second setup, the number of patches is variable in a range from 1 to 12. The fiber orientation is encoded with the same discrete parameter like in Section 7.1.2. Again, the first eigenfrequency $f_1$ is to be maximized under consideration of a mass constraint for the entire structure and a constraint limiting the number of empty elements $n_e$ to zero.

$$\begin{aligned} &\text{maximize} && f_1 \\ &\text{subject to} && m \leq 0.48275\ \text{kg} \\ & && n_e = 0. \end{aligned} \tag{7}$$

A mass constraint violation of 0.05 kg is tolerated but penalized in the fitness (maximal allowed mass:

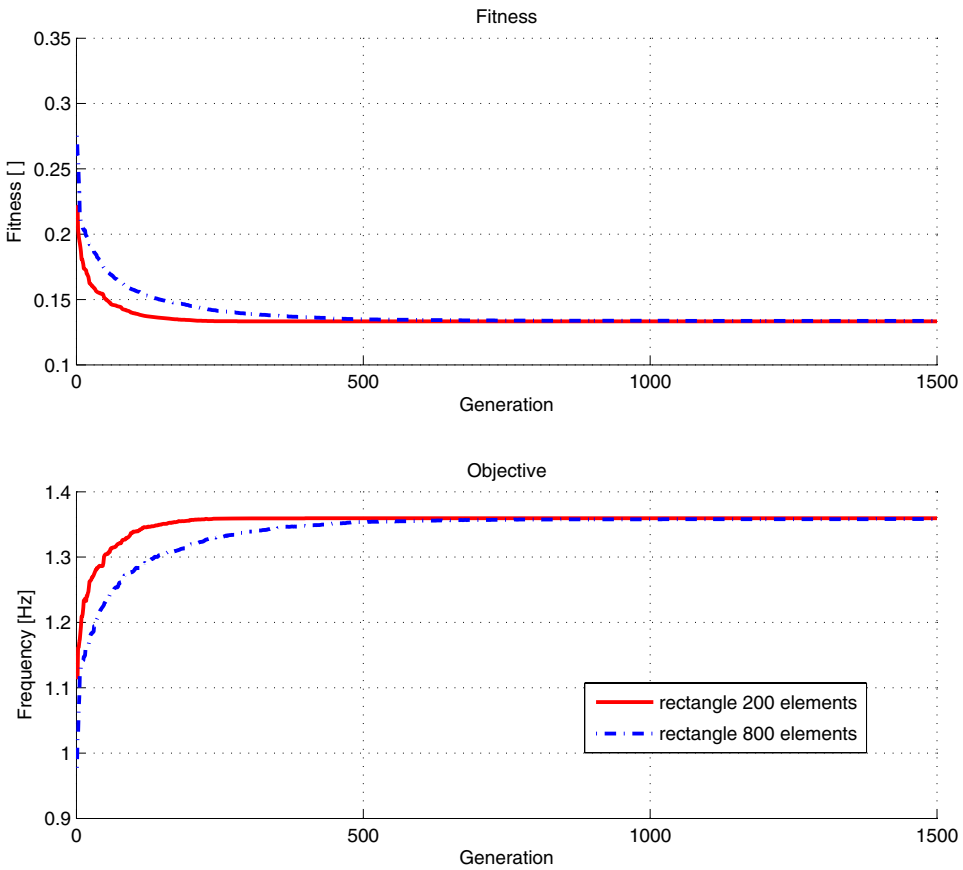



**Fig. 11** Convergence plots for the single patch rectangle optimization averaged over 12 runs

0.53275 kg). The number of elements with fill material should be zero; 20 are tolerated but again penalized.

**Results** The optimization is done in populations of 50 individuals over 10,000 generations with the 200 element model. The convergence plots (Fig. 13) show average fitness, objective, and constraint values of eight runs. The best result reaches an eigenfrequency of 27.8534 Hz with a mass of 0.500909 kg and without empty elements. The layers of the overall best solution with the highest first eigenfrequency can be seen in Fig. 12. Most of the material is concentrated around the mass. With a very stiff region around the mass, the first mode shape places higher amplitudes at the right half of the rectangle. Apparently, the sixth layer should avoid this effect by reinforcing the opposite of the mass region. Again, the patch shapes and, in particular, the ply angles do not clearly reflect the $x$-symmetry of the problem. From a manufacturing point of view, this solution is quite problematic, as a lot of complex patch shapes occur. This is caused by the relatively small discretization of the problem.

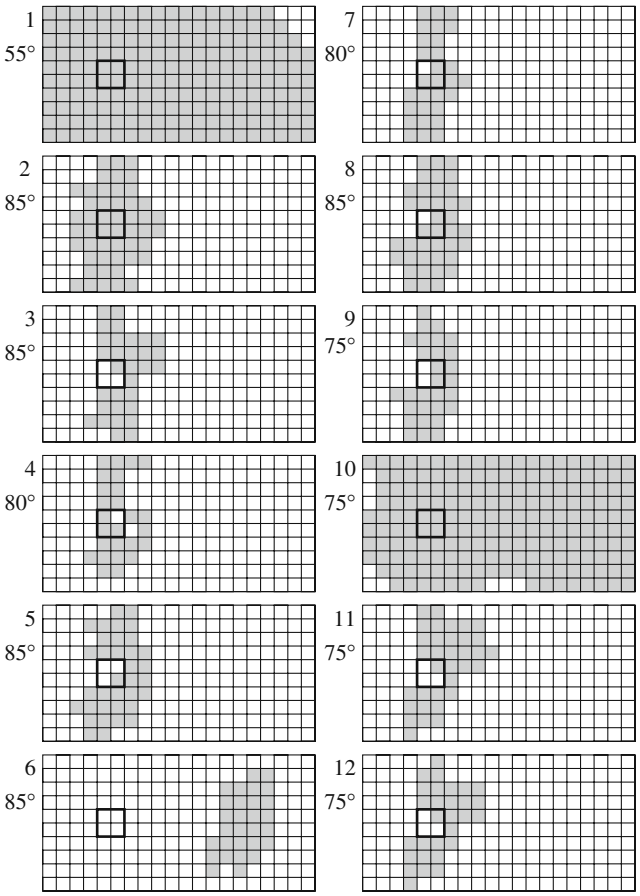



**Fig. 12** Best result of the constrained rectangle optimization problem: 12 layers in their stacking sequence from 1 to 12 with their ply angles

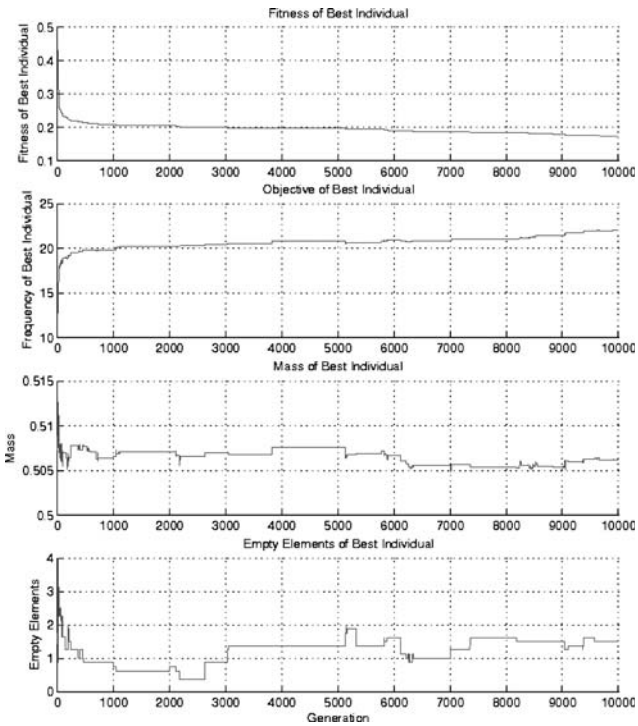



**Fig. 13** Convergence plots of constrained rectangle optimization averaged over eight runs

Figure 13 depicts a faster convergence for the first 1,000 generations than afterward. Only one of the eight analyzed runs reaches an eigenfrequency above 25 Hz. The large number of degrees of freedom in the parameterization combined with this type of fitness definition probably renders the problem multimodal. Therefore, it is assumed that most of the runs become stuck close to local optima. For such a simple optimization problem, there are obviously many degrees of freedom leading to an extremely large space of possible solutions. A possible improvement could be to introduce operators simplifying the patch shapes or grouping several finite elements to larger entities. Nevertheless, an unpublished comparative study using 12 rectangular patches (organized in a variable-length vector genotype) of the same UD material resulted in a maximum eigenfrequency of 25.0574 Hz by a comparable number of evaluations.

## 7.2 Minimal compliance design of an aircraft vertical tail

### *7.2.1 Problem*

Today's airplane vertical tail structures are completely made of composite material. Thus, the application of the newly introduced optimization method on a vertical

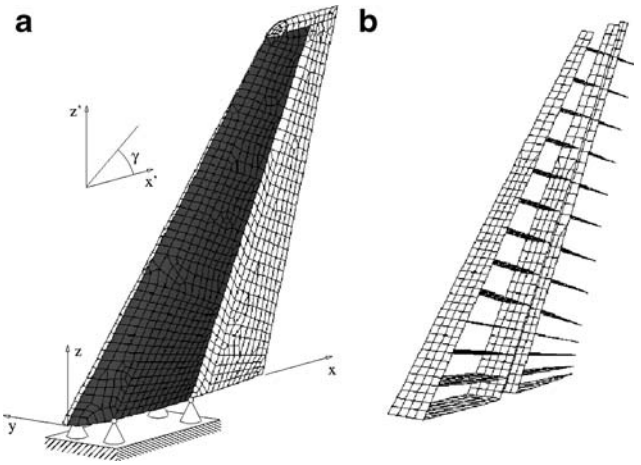



**Fig. 14** Vertical tail finite element model: **a** Model with boundary conditions and shaded optimization area. $\gamma$ is an exemplary ply angle. **b** Internal stringer and rip structure

tail allows to get an impression of its behavior on real engineering problems. The symmetric vertical tail (Fig. 14) is built of a stiff box containing 2 stringers and 13 rips. The two box stringers are attached to the fuselage. The rear part is formed of a less stiff rudder.

Four nodes of the box stringers connecting the tail-fin with the fuselage are modeled as support points ($u_{x,y,z} = 0$). A heavy side pressure load case of a turn maneuver is analyzed: Uniformly distributed forces in $y$-direction are applied on all nodes of the port side ($y \leq 0$). The bending and torsion of the structure lead to a maximal deformation $u_{\max}$ at the top rear corner of the rudder. A base laminate is applied to the whole structure. Local reinforcements should be added to increase the stiffness of the construction. The box and nose area carry most of the loads. Therefore, fiber-reinforced patches have to be placed there. The objective is to minimize the maximum deformation $u_{\max}$ at the top rear corner of the structure under an upper limit constraint for its mass $m_{\max}$ and a maximum number of patches $n_{\max}$.

$$\begin{aligned} &\text{minimize} && u_{\max} \\ &\text{subject to} && m \leq m_{\max} \\ & && n \leq n_{\max} \end{aligned} \tag{8}$$

Again, the fitness formulation is implemented according to König (2004) where the fitness of an individual is defined as a weighted sum of objective and mass constraint penalty terms. For the objective and the mass constraint, two mapping functions are used normalizing the evaluated results for the maximum deformation and the mass into an interval [0,1]. Thus, the weights of the weighted sum only reflect the relative importance of objective and constraint.

Some customization of the graph-based parameterization has to be done to meet the special needs of the problem. As the structure should remain symmetric to behave equally in right and left turns, patches are arranged only on one side and are then mirrored to the opposite side. Furthermore, the optimization region covers not the complete surface but only the box and the nose area as Fig. 14 depicts. These requirements are transferred to the genotype space with a reduced element graph that covers only the port side of the box and nose surface.

Each element takes a base layer (base laminate) before mapping. The optimization engine can choose from two different unidirectional materials (HTUD and HMUD) as shown in Table 2 to form additional patch layers in the optimization region. These materials are encoded with a discrete, unordered parameter (`string_gene` from the universal genotype by Konig 2004). Each patch has a discrete ply angle between −90° and +90° in steps of 5° (`const_float_list_gene`). A third integer parameter (`int_gene`) between one and 15 encodes the number of layers per patch. This parameter multiplies with the thickness of the respective material. So every patch vertex contains a representation of these three patch genes.

The optimization is done with a population size of 100 individuals, the global mutation rate is set to 0.4, and the global crossover rate is equal to 0.5, whereas the relative operator rates for all mutation and crossover operators are uniform. The maximum mass ($m_{\max}$) is set to 1,500 kg, and a variable number of patches in between one and eight ($n_{\max}$) are allowed. The optimization is stopped after 40 iterations without improvement.

### *7.2.2 Results*

The convergence curves of one optimization run are shown in Fig. 15. The best result after 250 generations reaches a maximum deformation of 174.902 mm and a mass of 1,509.77 kg. Figure 16 depicts the patches of the best solution ever found. Although the optimization is not fully converged, the result is quite convincing. It

**Table 2** Materials in the vertical tail optimization problem

| Material | HTUD | HMUD | Base | |
|---|---|---|---|---|
| $E_1$ | 135 | 220 | 55 | [GPa] |
| $E_2$ | 10 | 10 | 55 | [GPa] |
| $\nu$ | 0.3 | 0.3 | 0.04 | [–] |
| $G_{12}$ | 3.8 | 2.9 | 4.8 | [GPa] |
| $G_{23}$ | 5.0 | 5.0 | 2.5 | [GPa] |
| Thickness | 0.00015 | 0.00015 | 0.003 | [m] |
| Area density | 0.24 | 0.24 | 4.73 | [kg/m$^2$] |

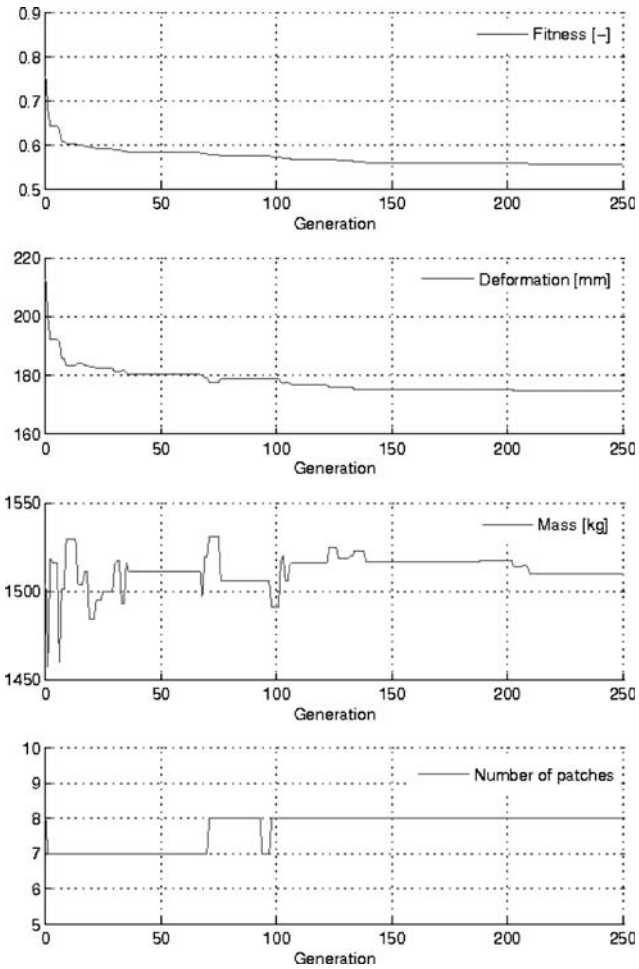



**Fig. 15** Convergence plots of the constrained vertical tail optimization problem

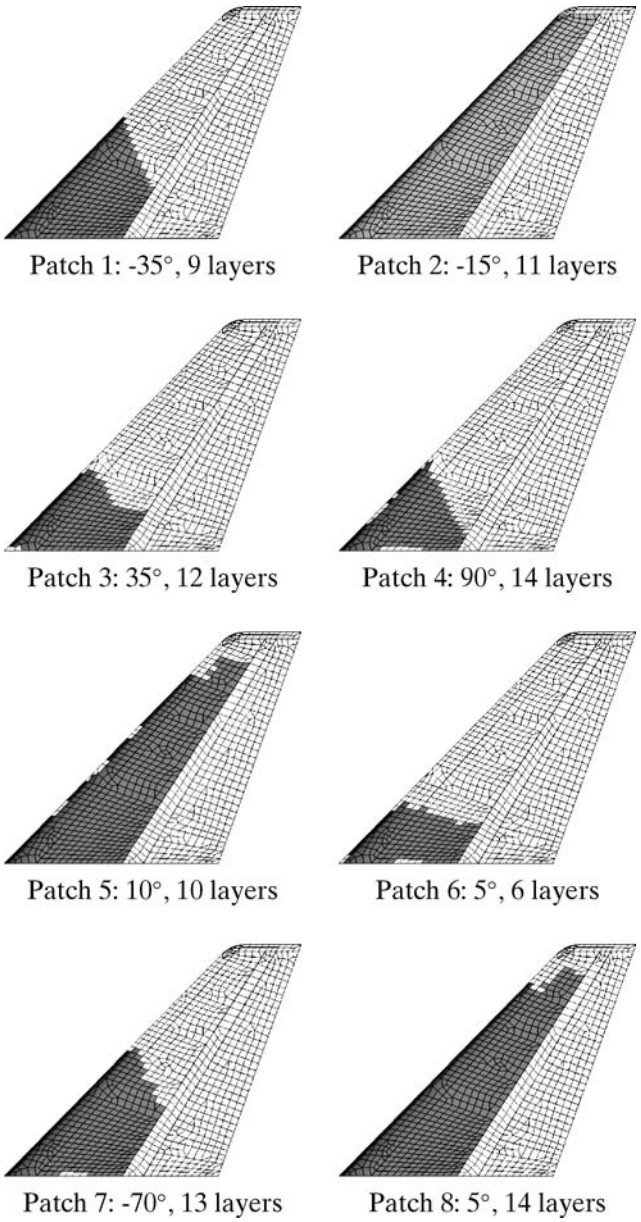



**Fig. 16** Best solution found for the vertical tail optimization problem

consists of eight layers of the material HMUD, which is not very surprising, as the higher Young's modulus of HMUD leads to a stiffer vertical tail. In case a stress or failure criteria constraint would be added to the optimization formulation (8), the high tension fibers would probably have been used as well. Figure 16 depicts a gradual decrease of laminate thickness toward the top of the vertical tail. All patch shapes completely fill the lower region of the vertical tail where they influence the maximum deformation the most. The top of the vertical tail has only little influence on the maximum deformation. Thus, only patch 2, which covers the entire optimization region, reaches the top of the tailfin. The patch thicknesses, i.e., their number of layers, reach values between 6 and 14 layers. The maximum number of 15 layers is not reached, but it seems to be more efficient to have a larger number of patches instead of a low number with the maximum number of layers.

Surprisingly, the ply angles cover a range from $-70°$ to 90°, which was not really expected, but this patch configuration seems to be superior to other stacking sequences with a dominating orientation direction.

Finally, the patch concept proves to be working for more complex optimization problems, although the degrees of freedom of the patch shapes are very large. Therefore, the optimization is quite expensive, maybe too expensive, as it is probably not necessary to allow completely arbitrary patch shapes. A further development of this method should therefore investigate a grouping of several finite elements to zones, where all finite elements within such a zone have the same stacking sequence. Moreover, the definition of these zones

should be used to introduce manufacturing knowledge and experience into the optimization setup.

## 8 Conclusion and outlook

A new graph-based parameterization for the evolutionary optimization of laminated composites structures is introduced. It is able to concurrently optimize size, shape, position, number, ply angle, and any other material-related property of fiber-reinforced patches. The applications demonstrate the methods ability to find good quality results in constrained optimization problems and on curved shells.

According to Goldberg (1989), a representation concept for evolutionary search should be complete, nonredundant, feasible, and preserving locality. These attributes are shortly discussed for the proposed representation: The presented representation concept is complete for composite structures consisting of simply connected areas; that is, for each phenotype solution, there exists at least one genotype solution. Furthermore, it is also nonredundant, so that, for each genotype solution, exactly one phenotype solution exists. The constraint-handling is discussed in terms of legality and feasibility. Illegal genotype configurations may represent noncohesive patches. These genotype solutions are avoided by initialization and variation operators preserving legality; that is, no illegal design solutions can occur. Infeasibility concerning external constraints implied by the fitness definition of the optimization task at hand cannot be directly incorporated into the representation concept but they are handled by a penalty method. This ensures the applicability of the method on a large range of optimization problems. Obviously, the feasible region of an optimization task is problem dependent and, in particular, dependent on the fitness formulation. It is therefore difficult to estimate the feasible region in the presented applications. However, it should be pointed out that, due to the variable dimensionality of the search space within one optimization run and even within a single generation, the commonly used methods to investigate and ensure feasibility, e.g., Michalewicz and Schoenauer (1996), do not apply to this approach; hence, further research is required. Finally, the variation operators are all designed to preserve locality.

The presented method requires further development. It turns out that the large number of degrees of freedom in the genotype space leads, when applied to engineering problems, to rough, multimodal fitness topologies. Adaptive fitness definitions that smooth constraint penalties out at the beginning of an optimization run as well as selective simplifications of the patch shapes during optimization in form of further developed variation operators could possibly speed up the convergence behavior. To reduce the number of degrees of freedom, one may think of grouping several finite elements to zones, whereas all these finite elements are assigned with identical properties. The number of discontinuities of orientations and thicknesses leading to stress risers could be reduced by introducing such zones. Furthermore, a stress/strain or failure criteria constraint should be included into the optimization formulation to eliminate highly stressed regions. Toward the optimal solution of laminate optimization problems, improvements seem to be influenced only by a small subset of variation operators that are able to perform slight and local changes. An adaption mechanism for operator rates that promotes successful variations could help to increase the performance of the method. Drapery problems are not yet discussed and require extensions of the method.

Following the concept of graph-based parameterizations on truss topologies by Giger and Ermanni (2006) and this new approach in laminate optimization, one may think of further applications of graph theory in computational structural optimization. The kind of laminate optimization bears a resemblance to mesh-based topology optimization where graph theory could give novel inputs as well.

## References

Giger M, Ermanni P (2006) Evolutionary truss topology optimization using a graph-based parameterization concept. Struct Multidisc Optim 32(4):313–326

Goldberg DE (1989) Genetic algorithms in search, optimization and machine learning. Addison-Wesley, Reading, MA

Grosset L, Venkataraman S, Haftka RT (2001) Genetic optimization of two-material composite laminates. Proceedings of the American Society of Composites—16th Annual Technical Conference, September 9–12, 2001, Blacksburg, VA

Hammer VB, Bendsoe MP, Lipton R, Pedersen P (1997) Parameterization in laminate design for optimal compliance. Int J Solid Struct 34(4):415–434

Huang J, Haftka RT (2005) Optimization of fiber orientations near a hole for increased load carrying capacity of composite laminates. Struct Multidisc Optim 30(5):335–341

Kane C, Schoenauer M (1996) Topological optimum design using genetic algorithms. Control Cybern 25(5):1059–1088

König OJ (2004) Evolutionary design optimization: tools and applications. Ph.D. Thesis, ETHZ

Le Riche RG, Knopf-Lenoir C, Haftka RT (1995) A segregated genetic algorithm for constrained structural optimization. In: Eshelman L (ed) Proceedings of the Sixth International Conference on Genetic Algorithms. Morgan Kaufmann, San Francisco, CA, pp 558–565

Michalewicz Z, Schoenauer M (1996) Evolutionary algorithms for constrained parameter optimization problems. Evolutionary Comput 4(1):1–32

Michalewicz Z, Xiao J, Trojanowski K (1996) Evolutionary computation: one project, many directions. In: International Syposium on Methodologies for Intelligent Systems, pp 189–201

Pedersen P (1989) On optimal orientation of orthotropic materials. Struct Optim 1:101–106

Pedersen P (1991) On thickness and orientational design with orthotropic materials. Struct Optim 3:69–78

Pedersen P (2004) Examples of density, orientation, and shape-optimal 2D-design for stiffness and/or strength with orthotropic materials. Struct Multidisc Optim 26:37–49

Siek JG, Lee LQ, Lumsdaine A (2002) The Boost Graph Library. C++ In-Depth Series. Addison Wesley

Venkataraman S, Haftka RT (1999) Optimization of composite panels—a review. In: Proceedings of the 14th Annual Technical Conference of the American Society of Composites Dayton

Zehnder N, Ermanni P (2006) A methodology for the global optimization of laminated composite structures. Elsevier Ltd., Composite Struct 72(3):311–320

Zienkiewicz OC, Taylor RL (2000) The finite element method, vol 2. Oxford, 5 edn, 2000. ISBN 0-7506-5055-9