Doctoral Thesis

Evolutionary design of laminated composite structures

Author(s):
Keller, David Christian

Publication Date:
2010

Permanent Link:
https://doi.org/10.3929/ethz-a-006082073

Rights / License:
In Copyright - Non-Commercial Use Permitted
Evolutionary Design of Laminated Composite Structures

A dissertation submitted to
ETH Zürich

for the degree of
Doctor of Sciences

presented by
DAVID CHRISTIAN KELLER
Dipl. Masch. Ing. ETH
born November 2, 1981
citizen of Mandach (AG) and Hottwil (AG)

accepted on recommendation of
Prof. Dr. P. Ermanni, examiner
Prof. Dr. Z. Gürdal, co-examiner
Prof. Dr. F. van Keulen, co-examiner
Dr. G. Kress, co-examiner

2010
Abstract

This dissertation investigates methods for the automated design and optimization of laminated composite structures. Optimal design of laminated composite structures is challenging due to the possibility to locally adapt the material system to the mechanical situation. Automated structural design on a computer is enabled by a combination of numerical simulation and optimization algorithms. The finite element method provides the possibility to predict mechanical properties of virtual candidate solutions. Numerical optimization algorithms then adapt the structure’s attributes in order to meet specific demands formulated on the aforementioned simulated properties. Evolutionary algorithms are a group of biologically inspired optimization algorithms which have repeatedly and successfully been applied to optimal design problems with laminated composites. This thesis focuses on methods to compose evolutionary algorithms for the specific traits of laminate optimization problems. A special focus is set on the variation state of a canonical evolutionary algorithm. This state is particularly influenced by the genetic representation of a candidate solution, i.e. the way the adjustable attributes are translated to machine readable entities. The aim of the thesis is to develop and examine genetic representation schemes to concurrently evolve a structure’s topology, shape, and laminate properties.

An overview of structural optimization and evolutionary computation illustrates the state-of-the-art. In variable-length representations, the dimensionality of the search space is a variable to optimize. The importance of variable-length representations in evolutionary topology and laminate optimization is exemplified. A weakness of established variable-length crossover operators is the treatment of length constraints. Based on existing concepts a split-and-splice variable-length crossover operator respecting length constraints is introduced.

In order to improve the solution quality of a real-encoded evolutionary algorithm, a gradient-based local search is embedded in the variation state of the algorithm. The algorithm intrinsic parallelization is extended to the variation state in order to cope with different runtimes of deterministic and stochastic operators. A parallelization of the variation state requires abandoning of synchronization points. Hence, the population is replaced by a pool of individuals where distributed breeder processes continuously draw samples for mating and insert offspring to replace parents. A lifetime concept is developed to keep the pool size approximately constant. A niching strategy focuses the stochastic component of the algorithm to unexplored regions.
of the search space. This hybrid, parallel, asynchronous evolutionary algorithm outperforms a conventional, sequential algorithm in test functions with a moderate number of local optima.

This algorithm is then applied on ply angle optimization problems. The ply angles are parameterized following the concept of global plies. Thus, the global cohesion of the structure is guaranteed. In simple academic benchmark problems the algorithm is capable to repeatedly find global optimal solutions. Two examples illustrate the applicability of the method to typical engineering problems.

A structured representation scheme aimed at the optimization of locally varying laminate properties on geometrically partitioned structures is proposed. It employs graph operations to generate connected reinforcement patches. Gradient information is used in genetic variation operations of the shape of reinforcements and ply angles. The method surpasses existing approaches in convergence and the results of repeated applications on the same problem show considerably lower spread. A case study illustrates a possible application of the method.

An arrangement-based representation for the optimization of topology attributes is investigated. The method employs a set of spline curves for the partition of the Euclidean plane into faces of different shape. An assignment of void and material on these faces then represents the structure’s topology. Graph operations are used to ensure a connected load path. The method is investigated in a cantilever problem and compared to a homogenization approach. Although a feasible result is achieved, the performance is not competitive. The robustness of geometric operations and automated meshing are identified as sources of inefficiencies. Experiments to avoid numerical difficulties in geometric operations by exact computation have been bring no remedy. Shape healing and repair algorithms and mesh checks are employed instead.

The arrangement-based representation is then simplified and an alternative meshing strategy enhances the robustness of the method. In order to allow for a combined laminate and topology optimization, the face properties are realized by the before introduced laminate optimization method. In a simple academic example the method outperforms the laminate optimization on a predefined partition of the structure. But the numerical requirements and the spread in the results is considerably increased. The same representation is extended by the possibility to optimize shape parameters as well. Therefore, the arrangement is projected on a parameterized shape. Numerical examples demonstrate the method’s ability to concurrently optimize shape and laminate properties. Although enabled by the representation, topological changes in the form of void regions are not observed. Dependencies between the shape
and topological level of the genetic encoding and inside the shape representation itself induce difficulties to create beam-like structural members.

The thesis investigates representation schemes along a path of increasing complexity. Hybrid and problem-specific variation operations are able to improve properties of existing methods despite of this increasing complexity. This is demonstrated in ply angle and laminate optimization problems. More complex genotype concepts reduce the number of optimization variables. At the same time the induce dependencies between these, which – like in the arrangement-based topology optimization – affect the solution of the problem. The arrangement-based encoding of shape attributes reveals not to be powerful enough for a combined topology and shape optimization of laminated composite structures.
Zusammenfassung


Um die Lösungsqualität eines reellwertig codierten evolutionären Algorithmus’ zu verbessern, wird eine gradientenbasierte, lokale Suche im Variationsschritt eingebettet. Die dem Algorithmus innenwährende Parallelität wird auf den Variationsschritt ausgeweitet um der unterschiedlichen Laufzeit deterministischer und stochastischer Operatoren gerecht zu werden. Die


Die Arrangement-basierte Repräsentation wird in der Folge vereinfacht und eine alternative Vernetzungsstrategie verbessert die Robustheit der Methode. Um die kombinierte Optimierung von Laminat und Topologie zu ermöglichen, werden Flächeneigenschaften mittels der zuvor entwickelten Laminatoptimierungsmethode erstellt. In einem einfachen, akademischen Beispiel

Acknowledgements

This thesis was accomplished at the Centre of Structure Technologies at ETH, Zürich. The presented work was supported by the Swiss National Science Foundation (SNF-project 200021-II 9689I1).

The following dedicative persons have contributed in various and important ways to this thesis. I would like to thank:

- Professor Paolo Ermanni and Dr. Gerald Kress for giving me the opportunity to work in this interesting field of research and for their advice.

- Professores Zafer Gürdal and Fred van Keulen for being co-examiners of my thesis.

- Dres. Mathias Giger, Oliver König, Nino Zehnder, and Marc Wintermantel for the inspiring discussions and for introducing me into the fascinating domain of structural optimization.

- All the people at the Centre of Structure Technologies, particularly my humorous office mates Dr. René Roos, Florian Hürlimann, Barbara Röhrnbauer, Florian Bachmann, and Benjamin Schläpfer.

- Michael Winkler, Benjamin Schläpfer, and Florian Bachmann for their comments and corrections to the manuscript.

- Martin Gamboni, Fabian Ehrich, and Wilfried Bürzle for their master’s and semester thesis contributing to this dissertation.

- My sister Ursula for her contributions on inequalities, for her comments to the manuscript, and for the constrained benchmark problems from particle physics.

- My parents and family for their lifelong support and encouragement.

April 2010, David Keller
## Contents

1 Introduction
   1.1 State-of-the-art ........................................ 2
      1.1.1 Topology and Shape Optimization .................. 2
      1.1.2 Laminate Optimization ............................. 9
      1.1.3 Combined topology, shape, and laminate optimization 16
   1.2 Motivation and Goals ................................. 17
   1.3 Thesis outline ....................................... 20

2 A canonical Evolutionary Algorithm ....................... 23
   2.1 Terminology ........................................... 23
      2.1.1 A general optimization problem ................... 23
      2.1.2 Classes of optimization algorithms ............... 24
      2.1.3 Evolutionary algorithms ........................... 24
   2.2 Components of an EA ................................ 27
   2.3 Representation concepts ............................... 29
   2.4 Fitness evaluation .................................. 32
      2.4.1 Separation method ............................... 34
      2.4.2 Exterior penalty method ......................... 35

3 A Split-and-Splice Variable-Length Crossover ............ 37
   3.1 Introduction ......................................... 37
   3.2 Incorporating length-constraints ...................... 41
   3.3 Split-and-splice crossover ............................ 42
   3.4 Concluding remarks ................................ 42

4 A Hybrid, Parallel, Asynchronous Evolutionary Algorithm
   Scheme .................................................... 45
   4.1 Introduction ......................................... 45
      4.1.1 Parallelization and asynchronism ................ 46
4.1.2 Speedup through hybridization ........................................ 51
4.1.3 Motivation and goals .................................................... 51
4.2 Method ........................................................................... 52
  4.2.1 A canonical non-linear optimization problem .................... 52
  4.2.2 Algorithm architecture .................................................. 52
  4.2.3 Maintaining genetic diversity ........................................ 54
  4.2.4 Life-cycle modeling and asynchronous selection ............... 55
  4.2.5 Initialization and asynchronous breeding ......................... 57
  4.2.6 Master process ............................................................ 58
  4.2.7 Hybridization by a deterministic variation operator .......... 59
  4.2.8 Exploiting niches ....................................................... 59
  4.2.9 Niche-based fitness sharing .......................................... 63
4.3 Implementation .................................................................... 64
4.4 Numerical experiments ....................................................... 64
4.5 Results and Discussion ....................................................... 66
4.6 Concluding remarks ........................................................ 73
4.7 Needs for further research ................................................ 74

5 Optimization of Ply Angles in Laminated Composite Structures 77
  5.1 Ply angle optimization: A non-linear optimization problem with periodic search space ........................................... 77
  5.2 Detailing the evolution engine ............................................ 78
    5.2.1 Arithmetic Crossover ................................................ 78
    5.2.2 Gaussian Mutation .................................................... 79
    5.2.3 SR1-mutation ............................................................ 82
  5.3 Implementation .................................................................. 82
  5.4 Numerical Experiments ..................................................... 82
    5.4.1 Minimal compliance design of a tensile specimen .......... 82
    5.4.2 Minimal compliance design of a rectangular plate ......... 85
    5.4.3 Eigenfrequency optimization of a circular disk ............. 87
  5.5 Applications ..................................................................... 87
    5.5.1 Minimum compliance design of an aircraft side-rudder .. 89
    5.5.2 Eigenfrequency optimization of a racing car rear wing ... 91
  5.6 Concluding remarks ........................................................ 94

6 Optimization of Laminated Composites on Geometrically Partitioned Shells 99
  6.1 Genetic representation and variation of laminated composites 99
    6.1.1 A laminate optimization problem formulated on sections 99
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1.2</td>
<td>Section categories</td>
<td>103</td>
</tr>
<tr>
<td>6.1.3</td>
<td>Section rankings with gradient information</td>
<td>103</td>
</tr>
<tr>
<td>6.1.4</td>
<td>Basic operations on the patch shape</td>
<td>104</td>
</tr>
<tr>
<td>6.1.5</td>
<td>Genetic variation operators</td>
<td>106</td>
</tr>
<tr>
<td>6.2</td>
<td>Implementation</td>
<td>108</td>
</tr>
<tr>
<td>6.3</td>
<td>Numerical Experiments</td>
<td>109</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Single-ply vibrating plate</td>
<td>109</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Twelve-ply vibrating plate</td>
<td>112</td>
</tr>
<tr>
<td>6.4</td>
<td>Application example: Mass optimization of an airplane side rudder</td>
<td>115</td>
</tr>
<tr>
<td>6.5</td>
<td>Concluding remarks</td>
<td>119</td>
</tr>
<tr>
<td>7</td>
<td>Genetic Encoding of Planar Spline Arrangements</td>
<td>121</td>
</tr>
<tr>
<td>7.1</td>
<td>Method</td>
<td>121</td>
</tr>
<tr>
<td>7.1.1</td>
<td>Planar arrangements</td>
<td>122</td>
</tr>
<tr>
<td>7.1.2</td>
<td>A topology optimization problem formulated on arrangements</td>
<td>122</td>
</tr>
<tr>
<td>7.1.3</td>
<td>Genetic encoding of planar arrangements</td>
<td>123</td>
</tr>
<tr>
<td>7.1.4</td>
<td>Genetic variation on arrangements</td>
<td>125</td>
</tr>
<tr>
<td>7.2</td>
<td>Implementation</td>
<td>132</td>
</tr>
<tr>
<td>7.3</td>
<td>Application example: Minimal compliance cantilever design</td>
<td>132</td>
</tr>
<tr>
<td>7.4</td>
<td>Comparison with a Homogenization Model</td>
<td>134</td>
</tr>
<tr>
<td>7.5</td>
<td>Conclusions</td>
<td>138</td>
</tr>
<tr>
<td>8</td>
<td>On the Robustness of Geometric Operations and Automated Meshing</td>
<td>141</td>
</tr>
<tr>
<td>8.1</td>
<td>Sources of geometric defects</td>
<td>141</td>
</tr>
<tr>
<td>8.2</td>
<td>Improving the robustness of geometric operations</td>
<td>143</td>
</tr>
<tr>
<td>8.2.1</td>
<td>Escaping to exact geometry</td>
<td>143</td>
</tr>
<tr>
<td>8.2.2</td>
<td>Shape healing and repair algorithms</td>
<td>146</td>
</tr>
<tr>
<td>8.3</td>
<td>Automatic meshing</td>
<td>147</td>
</tr>
<tr>
<td>8.3.1</td>
<td>Projection method</td>
<td>148</td>
</tr>
<tr>
<td>8.4</td>
<td>Concluding remarks</td>
<td>149</td>
</tr>
<tr>
<td>9</td>
<td>Arrangement-based Laminate Optimization on Structures with Variable Shape</td>
<td>151</td>
</tr>
<tr>
<td>9.1</td>
<td>Method</td>
<td>151</td>
</tr>
<tr>
<td>9.1.1</td>
<td>Genetic representation</td>
<td>152</td>
</tr>
<tr>
<td>9.1.2</td>
<td>Genetic variation operators</td>
<td>152</td>
</tr>
<tr>
<td>9.2</td>
<td>Numerical Experiments</td>
<td>155</td>
</tr>
<tr>
<td>9.2.1</td>
<td>Single-ply vibrating plate</td>
<td>155</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>9.2.2 Shape adaptive vibrating plate</td>
<td>157</td>
<td></td>
</tr>
<tr>
<td>9.2.3 Cantilever</td>
<td>161</td>
<td></td>
</tr>
<tr>
<td>9.3 Conclusions</td>
<td>166</td>
<td></td>
</tr>
<tr>
<td>10 Conclusions and Outlook</td>
<td>167</td>
<td></td>
</tr>
<tr>
<td>10.1 Concluding remarks</td>
<td>167</td>
<td></td>
</tr>
<tr>
<td>10.2 Outlook</td>
<td>170</td>
<td></td>
</tr>
<tr>
<td>10.2.1 Similarity evaluation by overlays</td>
<td>170</td>
<td></td>
</tr>
<tr>
<td>10.2.2 Manufacturing considerations</td>
<td>171</td>
<td></td>
</tr>
<tr>
<td>10.2.3 The surface concept</td>
<td>172</td>
<td></td>
</tr>
<tr>
<td>A Similarity-based Adaptive Fitness Sharing</td>
<td>i</td>
<td></td>
</tr>
<tr>
<td>A.1 Method</td>
<td>iv</td>
<td></td>
</tr>
<tr>
<td>A.1.1 Fitness Sharing Evolutionary Algorithm</td>
<td>iv</td>
<td></td>
</tr>
<tr>
<td>A.1.2 The Similarity Concept</td>
<td>v</td>
<td></td>
</tr>
<tr>
<td>A.1.3 Diversity Reward and Heuristic Feedback-Control of Niche Radius</td>
<td>vi</td>
<td></td>
</tr>
<tr>
<td>A.2 Validation</td>
<td>viii</td>
<td></td>
</tr>
<tr>
<td>A.2.1 Method</td>
<td>viii</td>
<td></td>
</tr>
<tr>
<td>A.2.2 Results</td>
<td>ix</td>
<td></td>
</tr>
<tr>
<td>A.3 Conclusions</td>
<td>xii</td>
<td></td>
</tr>
<tr>
<td>B Material Data</td>
<td>xv</td>
<td></td>
</tr>
<tr>
<td>Bibliography</td>
<td>xvii</td>
<td></td>
</tr>
<tr>
<td>Curriculum Vitae</td>
<td>xlv</td>
<td></td>
</tr>
</tbody>
</table>
List of Figures

1.1 Optimization disciplines ................................... 3
1.2 Classes of parameterization concepts in TOD .............. 4
1.3 Representation concept based on stretched and clipped cubes 7
1.4 Lamination zones ............................................. 13
1.5 Mapping of patches to laminate sections ................. 14
1.6 CAD-based patches ........................................... 15
1.7 Geometry-based patches ..................................... 15
1.8 Graph-based patches ......................................... 16
1.9 Zonal graph-based patches .................................. 17

3.1 The inter-species crossover problem ....................... 39

4.1 Classification of parallel evolutionary algorithms .......... 46
4.2 Algorithm structures for different parallel EAs .......... 48
4.3 Asynchronous, parallel optimization architecture .......... 52
4.4 Unidirectional circular migration scheme with \( d \) demes. . 55
4.5 Results for the Sphere test function ....................... 68
4.6 Results for the Rosenbrock test function ................... 69
4.7 Results for the Michalewicz test function .................. 70
4.8 Results for the Schwefel test function ..................... 71
4.9 Results for the Rastrigin test function ..................... 72

5.1 Arithmetic crossover with periodic box-constraints ........ 80
5.2 Gaussian mutation with periodic box-constraints .......... 81
5.3 Geometry and objective of the tensile specimen experiment. 83
5.4 Convergence plot for numerical experiments ............. 84
5.5 Geometry and objective of the rectangular plate experiment. 86
5.6 Contour plot of the objective for the rectangular plate experiment ........................................ 88
<table>
<thead>
<tr>
<th></th>
<th>Figure Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.7</td>
<td>Aircraft side rudder structure</td>
<td>90</td>
</tr>
<tr>
<td>5.8</td>
<td>Laminate on the aircraft side rudder</td>
<td>91</td>
</tr>
<tr>
<td>5.9</td>
<td>Convergence plot for the side rudder optimization</td>
<td>92</td>
</tr>
<tr>
<td>5.10</td>
<td>Racing car rear wing structure</td>
<td>93</td>
</tr>
<tr>
<td>5.11</td>
<td>Racing car rear wing laminate</td>
<td>94</td>
</tr>
<tr>
<td>5.12</td>
<td>Convergence plot for the rear wing optimization</td>
<td>95</td>
</tr>
<tr>
<td>6.1</td>
<td>Geometry and corresponding section-graph of a curved panel partitioned to six sections.</td>
<td>100</td>
</tr>
<tr>
<td>6.2</td>
<td>Illustration of different types of sections in a structure with non-regular sections and a single patch.</td>
<td>102</td>
</tr>
<tr>
<td>6.3</td>
<td>Crossover pairings for parents of different length in Alg. 18</td>
<td>109</td>
</tr>
<tr>
<td>6.4</td>
<td>Geometry of the vibrating plate experiment</td>
<td>110</td>
</tr>
<tr>
<td>6.5</td>
<td>Results of the single-ply vibrating plate experiment</td>
<td>110</td>
</tr>
<tr>
<td>6.6</td>
<td>Best individuals identified in 10 different runs for the single-ply vibrating plate experiment</td>
<td>111</td>
</tr>
<tr>
<td>6.7</td>
<td>Convergence plots for the twelve-ply vibrating plate experiment</td>
<td>113</td>
</tr>
<tr>
<td>6.8</td>
<td>Plies of the best individual for the twelve-ply vibrating plate experiment</td>
<td>114</td>
</tr>
<tr>
<td>6.9</td>
<td>Aircraft side rudder with reference points A, B, and C.</td>
<td>115</td>
</tr>
<tr>
<td>6.10</td>
<td>Skin of the side rudder partitioned to 36 lamination sections</td>
<td>117</td>
</tr>
<tr>
<td>6.11</td>
<td>Convergence plots for the side rudder application</td>
<td>118</td>
</tr>
<tr>
<td>6.12</td>
<td>Plies of the side rudder optimization application</td>
<td>119</td>
</tr>
<tr>
<td>7.1</td>
<td>Topology optimization problem</td>
<td>123</td>
</tr>
<tr>
<td>7.2</td>
<td>Graph-representations in planar arrangements.</td>
<td>124</td>
</tr>
<tr>
<td>7.3</td>
<td>Design space Ω, supports and loading for the minimal compliance cantilever problem.</td>
<td>133</td>
</tr>
<tr>
<td>7.4</td>
<td>Best solution found for the minimal compliance cantilever problem.</td>
<td>134</td>
</tr>
<tr>
<td>7.5</td>
<td>Convergence plots for the cantilever application</td>
<td>135</td>
</tr>
<tr>
<td>7.6</td>
<td>Optimum design of the cantilever problem found by a homogenization method</td>
<td>136</td>
</tr>
<tr>
<td>8.1</td>
<td>Approximative intersection of two splines</td>
<td>143</td>
</tr>
<tr>
<td>8.2</td>
<td>Approximation of a spline by a polyline</td>
<td>144</td>
</tr>
<tr>
<td>8.3</td>
<td>Influence of non-phenotypic edges on automated mesh quality</td>
<td>149</td>
</tr>
<tr>
<td>9.1</td>
<td>Hierarchical representation of laminate sections on a structure of variable shape</td>
<td>153</td>
</tr>
<tr>
<td>9.2</td>
<td>Free edges induced by curves with arbitrary endpoints</td>
<td>155</td>
</tr>
</tbody>
</table>
9.3 Representation of void includes (bubbles) by a single closed curve compared to two bound curves .............................. 156
9.4 Results of the single-ply vibrating plate experiment ............ 157
9.5 Best individuals identified in 10 different runs for the single-ply vibrating plate experiment ........................................ 158
9.6 Shape adaptive vibrating plate experiment .......................... 159
9.7 Convergence plots for the shape adaptive vibrating plate ex- experiment ................................................................. 159
9.8 Shape of the best solution for the shape adaptive vibrating plate experiment ............................................................. 160
9.9 Patches of the best solution for the shape adaptive vibrating plate experiment ............................................................. 160
9.10 Contour plot of the displacement field for the first mode in the best solution for the shape adaptive vibrating plate experiment 161
9.11 Shape adaptive cantilever geometry ................................. 163
9.12 Convergence plots for the shape adaptive cantilever experiment 164
9.13 Patches and shape of the best solution found for the shape adaptive cantilever problem ................................................. 165

10.1 Similarity evaluation by overlays ................................. 171
10.2 Groundstructure of a stringer reinforced panel .................. 172
10.3 Representation of a stringer reinforced panel topology ........ 173
List of Tables

1.1 Overview of evolutionary laminate optimization methods with problem-specific representation. ............................ 18

4.1 Test functions used in numerical experiments ............... 65
4.2 Lowest fitness value for each function and each configuration 67

5.1 Best solutions and detected niches for the tensile specimen experiment ......................................................... 85
5.2 Best solutions for the rectangular plate experiment ........ 87
5.3 Best solutions for the disk experiment ........................... 88

A.1 Unconstrained (F) and constrained (C) benchmark problems with search space $\mathbb{X}$ and best known solution $F(x^*)$. All objectives are to be minimized. ................................. xi
A.2 $p$-values of Wilcoxon rank-sum tests for the three configurations NS, FS, and AFS ............................... xiii

B.1 Homogenized, orthotropic mechanical properties of a selection of composite materials ............................. xvi
## Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAD</td>
<td>Computer Aided Design. 6, 10, 100, 111, 138</td>
<td></td>
</tr>
<tr>
<td>CAE</td>
<td>Computer Aided Engineering. 100, 111, 113</td>
<td></td>
</tr>
<tr>
<td>CLT</td>
<td>Classical Laminate Theory. 7</td>
<td></td>
</tr>
<tr>
<td>CMA-ES</td>
<td>Covariance Matrix Adaptation Evolution Strategy. ii, 16</td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit. 32, 33, 45</td>
<td></td>
</tr>
<tr>
<td>DBSCAN</td>
<td>Density-Based Spatial Clustering of Applications with Noise. iii</td>
<td></td>
</tr>
<tr>
<td>DCEL</td>
<td>Doubly Connected Edge-List. 99</td>
<td></td>
</tr>
<tr>
<td>DMO</td>
<td>Discrete Material Optimization. 9, 10</td>
<td></td>
</tr>
<tr>
<td>EA</td>
<td>Evolutionary Algorithm. iii, 6, 8–10, 13, 16–18, 21, 31–33, 35–37, 39, 42, 43, 45–47, 54, 55, 58, 74, 75, 80, 81, 97, 100, 101, 111, 112, 121, 137, 140</td>
<td></td>
</tr>
<tr>
<td>ECP</td>
<td>Element Connectivity Parameterization. 5</td>
<td></td>
</tr>
<tr>
<td>EP</td>
<td>Evolutionary Programming. 16</td>
<td></td>
</tr>
<tr>
<td>ES</td>
<td>Evolution Strategy. 16</td>
<td></td>
</tr>
<tr>
<td>ESO</td>
<td>Evolutionary Structural Optimization. 6</td>
<td></td>
</tr>
<tr>
<td>Acronym</td>
<td>Definition</td>
<td>Pages</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------------------------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>FEM</td>
<td>Finite Element Method. 2, 118</td>
<td></td>
</tr>
<tr>
<td>FMA</td>
<td>Free Material Approach. 9</td>
<td></td>
</tr>
<tr>
<td>GA</td>
<td>Genetic Algorithm. 16, 17, 19, 37, 54, 138</td>
<td></td>
</tr>
<tr>
<td>GP</td>
<td>Genetic Programming. 17</td>
<td></td>
</tr>
<tr>
<td>HDM</td>
<td>Homogenization Design Method. 2, 5</td>
<td></td>
</tr>
<tr>
<td>MP</td>
<td>Mathematical Programming. 15, 16, 18, 31, 33</td>
<td></td>
</tr>
<tr>
<td>NURBS</td>
<td>Non-Uniform Rational B-Spline. 6</td>
<td></td>
</tr>
<tr>
<td>SIMP</td>
<td>Solid Isotropic Material with Penalization. 5, 9</td>
<td></td>
</tr>
<tr>
<td>SMT</td>
<td>Shear-Moment-and-Torque. 69</td>
<td></td>
</tr>
<tr>
<td>SR1</td>
<td>Symmetric Rank-1 Update. 42, 43, 45–47, 54, 60, 62, 64, 70, 72</td>
<td></td>
</tr>
<tr>
<td>SUS</td>
<td>Stochastic Universal Sampling. iii, 19, 97</td>
<td></td>
</tr>
<tr>
<td>TOD</td>
<td>Topological Optimum Design. 2, 5, 6, 23, 25, 97, 98, 112, 138, 139</td>
<td></td>
</tr>
<tr>
<td>TRIZ</td>
<td>theory of inventive problem solving (romanized from Russian: Teoriya Resheniya Izobretatelskikh Zadatch). 21</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

This dissertation investigates methods and aspects in the field of structural optimization. It responds to a worldwide interest in the automated design of structural components made from laminated composites. During the last decades the usage of fiber reinforced plastics in aero-space, transportation, and sport applications has increased fastly. The industrial interest in these material systems has driven efforts to establish numerical simulation methods and tools to predict the mechanical behavior of composite structures. The need for cumbersome design iterations by human designers has been reduced through the availability of optimization strategies employing these simulation methods to solve optimum design problems formulated on the predicted qualities of a desired product. The demand for even more efficient, autonomous, universal, and easy-to-use optimization strategies is still growing with the decreasing time-to-market cycles of nowadays products. Local load paths and hence mechanical properties of a structure are determined by its topology, shape, and material attributes. The additional degrees of freedom in design offered by laminated composites allow for a custom tailoring of the structural response of a final product but at the same time increase the complexity of finding optimal designs. This thesis aims at fusing approaches to determine optimal topology, shape, and material attributes in a single optimization strategy. The focus is set on aspects related to the parameterization, also called representation, of the structures attributes to be optimized.
1.1 State-of-the-art

Each approach to automated structural optimization features three components [74]: an optimization model, a structural model, and an optimization algorithm. The optimization model is based on a formulation of the optimization problem and a parameterization or representation scheme. The optimization problem describes target product qualities, usually in the form of one or more objectives and a set of constraints. The representation scheme consists of a choice of decision variables $x$ in a search space $X$ characterizing attributes to be optimized. The structural model reflects the physical behavior relevant to predict the product qualities driving the optimization objectives and constraint values. In the here considered optimization of laminated composites, the models employed are governed by mechanical laws. Closed form solutions are typically only available for simple geometries and loads. The Finite Element Method (FEM) (e.g. [267]) is an established approach to find approximate solutions even for complex geometries and loadings. Finally, the role of the optimization algorithm is to identify one or more choices for the decision variables $x$ optimizing the objectives and fulfilling all constraints. If the objectives and constraint values have to be computed by numerical simulations for each choice of decision variables, there is usually no possibility to retrieve a direct solution to the optimization problem. Hence, most approaches rely on iterative, numerical optimization algorithms searching for approximate solutions.

One can discern four major disciplines in the field of automated structural optimization (Fig. 1.1): Topology optimization or Topological Optimum Design (TOD) is concerned with finding an optimal material distribution within a given domain. It is related to the global layout of structural members (or holes), i.e. their number, position, and connectivity. Topology optimization often involves the shape. Shape optimization is seeking for an optimal contour or shape of a structural system without changing its topology. Sizing optimization addresses the search for optimal dimensions (e.g. cross-sections of truss members) of a system whose topology and shape is fixed. Material optimization finally aims at finding optimal local material attributes within a structure. In the following, the parameterization related aspects of established methods in these disciplines shall be outlined.

1.1.1 Topology and Shape Optimization

The here proposed classification of TOD methods into three main classes, i.e. voxel-based, component-based, and boundary-based, uses the parameterization (or representation, see Sec. 2.3) of the optimization problem as the
1.1 State-of-the-art

(a) Original structure

(b) Topology optimization

(c) Shape optimization

(d) Sizing optimization

(e) Material optimization

Figure 1.1: Optimization disciplines illustrated on a truss structure.
Figure 1.2: Classes of parameterization concepts in TOD
only feature to discriminate classes (Fig. 1.2). It aims at emphasizing similarities of approaches emerging from different branches of research in structural optimization. Other classifications based on the optimization algorithms are equally possible and can be extracted from the surveys presented by Saitou et al. [216], Eschenauer and Olhoff [76], Roy et al. [214], and Mackerle [167].

### Voxel-based methods

The most popular approaches to TOD can be found in the class of *voxel-based* methods. Bendsøe and Kikuchi [36] pioneered a group of optimization techniques referred to as homogenization methods (Homogenization Design Method (HDM)). There, the domain is partitioned into small finite cells (voxels, elements). An optimization strategy assigns material or void to each one of these cells. A strong coupling to the numerical model describing the structural behavior by means of the FEM characterizes these methods. The domain is filled with a porous composite material consisting of microscopic cells of the massive material and voids. Geometric parameters describing this microstructure then allow for variable stiffness, mass, and even anisotropy properties in each cell. The name HDM refers to cell-wise homogenization of the microstructure’s properties for in the evaluation step.

Bendsøe [33, 35] also proposed a variant of HDM referred to as Solid Isotropic Material with Penalization (SIMP). The originally discrete assignment problem is rendered continuous by introducing a smooth density function for each element allowing for intermediate states in the range void-material. The elastic properties of the material are coupled to the density by power law functions. A penalization is engaged to arrive at the desired void-material distribution.

Although these methods initially addressed the convex and differentiable minimum compliance problem with a single mass constraint, several extensions made them applicable on functionally constrained optimization problems [55], frequency optimization [70, 166, 251], optimization with anisotropic material models [34, 144], multi-material problems [31], crash worthiness problems [176], reliability optimization [136, 130], and shape optimization with smooth spline boundaries [175]. A detailed description of the methods and applications can be taken from a textbook by Bendsøe and Sigmund [37]. Rozvany [215] gives a survey and history of SIMP, HDM, and some derived approaches.

Similar parameterizations can be found in various methods employing local optimality criteria instead of a global objective like e.g. [27, 169, 173, 174]). Local optimality criteria are predestinated and successfully applied to solve optimization problems with local constraints like strength. They are often
inspired by an overall goal of maximum strength, fully stressed design, or minimum weight with strength constraints. The role and balance of different contributions driving these criteria is however not exposed to the user of such a method. Thus, exact target properties for a structure are difficult to reach.

Methods to directly address the binary decision space of voxel-based TOD problems by heuristic algorithms have been proposed, like e.g. [131, 127, 259, 255, 161]. Such methods eliminate the need for intermediate states in void-material. Global search algorithms make them applicable to non-convex problems. However, they are in general not competitive to HDM or SIMP in convex problems.

Instead of a parameterization of the element stiffness, the parameterization of element connectivities has been proposed by [140, 152]. These methods (called Element Connectivity Parameterization (ECP)) operate on virtual, zero-length elastic links connecting the otherwise unchanged elements. A sizing strategy is then employed to determine an optimal stiffness for each link. This methods is particularly relevant if large deformations have to be considered and SIMP-related approaches would therefore experience numerical difficulties.

**Component-based methods**

*Component-based* representations arrange parameterized, primitive structural members like bars, beams, or panels in the design space to solve a TOD problem. The so-called ground structure approach is most often applied to the optimization of truss topologies [32, 141]. There, a given set of nodes is connected with trusses with varying cross sections. In a first approach, the location of the nodes is fixed (e.g. [100]). Since the number of structural members remains the same during the optimization and only sizing parameters change, the design space can be continuous and is therefore suitable for mathematical programming methods. Further developments of the ground structure approach allow movable nodes (e.g. [253, 101]) and are hence capable to solve a shape optimization problem as well: A combination which is commonly referred to as layout optimization. Pedersen and Nielsen [193] applied the ground structure approach to the optimization of trusses subject to a variety of constraints. The connectivity and maximal number of all members is fixed within the given ground structure. Thus, only a solution realizable within the given ground structure can be found. This restriction offers the possibility to incorporate prior knowledge or manufacturing constraints. At the same time it introduces a bias in the search for creative designs.
Bentley and Wakefield [42] pioneered a representation concept which is in principle independent from the structural simulation method: They proposed a set of primitive polyhedra as base components of a spatial partitioning system. A layout problem of optical prisms is successfully tackled with this approach [43, 40] (Fig. 1.3). In [39] the same author identifies component-based representations as a key to employ Evolutionary Algorithms (EAs) as exploring problem solvers instead of optimizers of existing solutions, thus leading to more innovative and creative designs. The methodology is closely linked to a class of optimization algorithms, i.e. EAs. Further efforts have been made by various researchers towards more abstraction in genotype representations of structural systems. In the class of variable-length representations the number of decision variables is no longer fixed by a variable to be optimized. Hence, variable-length representations allow the algorithm to evolve solutions with a varying number of entities. The algorithm evolves the dimensionality of the search space as well as the actual solutions (e.g. [205]). Graph-representations have been developed for EAs [91, 254, 218]. An emerging category of representations builds on the parameterization of purely functional building rules of a structure instead of its attributes. Such representations are cellular au-
tomata (e.g. [108]), Lindenmayer-systems (e.g. [58]), mathematical graphs or trees (e.g. [221]), and embryogenies (e.g. [41, 121]). The survey of Kicinger et al. [138] presents a structured overview of 304 publications and a table of recently published representation concepts within this context.

Recently, component-based representations gained attention for structural optimization within Computer Aided Design (CAD)-environments. Although usually not referred to this exact term, they are closely related to the above outlined embryogenetic approaches. Today’s common CAD systems operate on parameterized chains consisting of functional operators (features) and relations connecting and ordering them (associations) (see e.g. [155]). Approaches to the encoding of features in TOD can be found for instance in [177] or [143]. A more advanced concept for the representation of feature-trees is first proposed in [89] and generalized in [257]. These approaches eliminate at least partially the need for cumbersome post-processing steps since they generate high quality geometric models of optimal designs. Furthermore, they allow the incorporation of prior knowledge, complex geometric design rules, or manufacturing considerations on the level of a geometric model. Geometric operations can be numerically expensive and non-robust.

**Boundary-based methods**

The class of *boundary-based* representation concepts in TOD is influenced by shape optimization methods: i.e. most TOD approaches in this category can be interpreted as generalizations of parametric shape optimization approaches. There, the boundary curves or surfaces are represented by parameterized geometric entities like, e.g., splines [48]. Topological changes may be induced by allowing the shape parameters to change drastically or by introducing or removing boundary components. A prominent approach realizing the later idea is known as the bubble-method proposed by Eschenauer et al. [75]. This approach parameterizes the boundary of void regions – so called bubbles – inside a monolithic domain. It starts an iterative sequence of shape optimizations and insertion of new bubbles. Thus, the dimensionality of the search space is changed upon the insertion of a new bubble. Similar methods can be found in the field of Evolutionary Structural Optimization (ESO). Cervera and Trevelyan [52, 53] introduce a boundary-based representation concept on Non-Uniform Rational B-Spline (NURBS). There, the introduction of internal NURBS-rings allows for topological changes.

The family of level-set methods has gained much attention during the last years, e.g. in [12, 13, 54, 187, 227, 261]. There, the boundary of the structure is represented as a level-set of a parameterized, higher-dimensional function.
Common functions are e.g. synthesized from radial basis-functions. Parametric variations of this smooth level-set function can then induce topological changes.

Boundary-based methods show a considerably smaller number of decision variables when compared to voxel-based methods. Their results need less post-processing, since the geometric modelling is part of the optimization itself. There is initially no direct coupling of the optimization and structural simulation method. If FEM is employed, this may require remeshing of the candidate solution before evaluation. Numerical differentiation is more challenging since the decision variables are no longer element specific.

Combined approaches

There are methods incorporating traits of more than one of the above introduced categories: A voxel- and boundary-based approach is presented by Tai and Chee [238] and Wang and Tai [254]. Parameterized curves in the design space form the skeleton of a structure. These curves are then projected onto a fixed grid of cells. According to some rules further flesh material in the form of additional cells is added around the skeleton. Thus, this method separates topology and shape optimization. Other connections between voxel- and boundary-based approaches can be found in the domain of post-processing methodologies like e.g. [123]. An approach called Gaussian Snowballs incorporating voxel- and component-based characteristics is presented by Wintermantel [258]. There, a set of fuzzy, parameterized components – called Gaussian Snowballs – is used as building blocks for a structure. Hamda et al. [110] evolve voxel-based representations to Voronoi or fractal abstractions with component-based traits.

1.1.2 Laminate Optimization

Composite materials are combinations of at least two materials. In the here considered class of fiber reinforced composites there are usually two components on a microscopic level: a polymer matrix and a fibrous reinforcement material. The matrix component supports the fibers in a given position, transfers and introduces loads into and between them, and protects them from environmental influences. The fibrous component contributes to an increased stiffness and strength of the composite. The choice of a material system is closely related to the manufacturing process. For the here considered, laminated composite structures, reinforcements are often prefabricated, woven fabrics. In contrast to the placement of single filaments, this simplifies handling and finally allows for manufacturing of larger structural components in
an affordable time. However, it restricts the freedom in designing composite structures.

Reinforcement fabrics exhibit preference directions depending on their weave. Unidirectional reinforcements take this to an extreme with parallel warp yarns with structural properties and negligible – ideally in existent – filling yarns. These preference directions become evident in the mechanical properties of the composite material, e.g. in its anisotropic stiffness, strength, or thermal expansion properties. Because there are significant differences in the material behavior between the main reinforcement direction (i.e. the fiber direction in a unidirectional ply) and transverse directions, several plies of potentially different fabrics and orientations – then called laminae – are usually stacked.

The multi-scale construction of fiber reinforced composites has to be considered in its mechanical modeling: following Classical Laminate Theory (CLT) (e.g. in [129, 224]), the mechanical properties of the fibrous and the matrix component are homogenized for each lamina. The laminate then, features new homogenized mechanical properties which can be assembled from the single lamina’s properties. Each lamina’s mechanical properties (and hence the material choice), its orientation, its thickness, and its position in the laminate stack contribute to the homogenized properties and hence characterize the laminate. Laminate optimization is carried out to optimize a subset or all of these variables.

**Ply angle optimization**

The orientation of each lamina or ply in the laminate stack allows for a mass-neutral tuning of the mechanical properties of the final product. Even for existing designs the optimization of ply angles can be useful because there is typically only little influence on manufacturing.

The assembly of laminate properties requires a rotation of the properties of each single lamina to a global laminate coordinate system. This rotation involves trigonometric functions. Hence, objectives building on top of mechanical properties of laminates are non-convex with respect to ply angles.

Analytical approaches to the optimization of ply angles (like e.g. [202, 194, 195, 72, 137, 249]) are only available for geometrically simple cases. The application of mathematical programming either risks to get stuck in a local optimal solution (which may be accepted in some situations, like e.g. [245, 128]) or requires extensive tuning of the optimizer itself in order to converge in the global optimal solution (e.g. [180, 49]). So far, a transformation of the original to a convex optimization problem e.g. by lamination parameters is not available for general shell structures but only for simple geometries.
1.1 State-of-the-art

([102, 103, 112, 179, 82, 3]). Furthermore, the back-transformation from once found optimal lamination parameters to a real lay-up is a non-trivial task and may require a second optimization step (cf. [16, 118]).

Although numerically expensive, evolutionary approaches have been applied to ply angle optimization (e.g. [106, 252, 183]). In order to reduce the computational requirements of a global search by EAs an iterative procedure including a local gradient-based search is proposed by [124].

Usually, the ply angle is considered to be constant within pieces of fabric. However, manufacturing processes like tow placement can achieve locally varying fiber angles. An optimization method exploiting this possibility is proposed by [47].

Optimization of the laminate stack

Combinations of ply angle optimization with the optimization of ply materials (cf. e.g. [153, 154]) or the optimization of ply thicknesses (like [83, 189]) lead to problems with partially discrete search space. This is particularly relevant, if practical constraints like e.g. induced by the manufacturing process have to be considered. Such constraints may include discrete ply angles or restricted ply ordering. Stochastic optimizers like EAs are employed there [4, 160, 183, 7].

For in-plane loadings the laminate stiffness and strength depends only on the fiber orientations, thicknesses, and elastic properties of the single laminae. Out-of-plane (e.g. bending, buckling, or frequency) situations involve the position of each lamina in the laminate stack as a further decision variable.

Apart from mathematical programming (e.g. [246, 79]), stochastic optimizers have repeatedly and successfully been applied to the optimization of the laminate stack: These include variants of EAs [142, 231, 243, 241, 158, 242, 206, 197, 56, 139, 192, 196, 29], heuristic branch-and-bound methods [172, 244], tabu-search [191], particle swarm optimization [237, 189], or ant-colony optimization [17]. A short overview over different parameterization schemes and associated optimization methods for laminate optimization can be found in [250].

Optimization of locally varying laminate properties

The above methods focus on the optimization of structures built from one single laminate. However, an advantage of laminated composites is the possibility to tailor their properties to the local requirements at different positions in the structure. Thus, to fully exploit the potential of composite materials,
locally varying laminates have to be considered. Laminate optimization problems accounting for locally varying laminates directly address the problem of how to distribute material in a design space. Thus, they can be interpreted as a fusion of topology and material optimization. This becomes evident if \textit{void} is accepted as an additional, virtual material as well.

Given the link to topology optimization, it is not astonishing that the most far going methods in terms of exploiting the freedom in design in this field are derived from homogenization methods and introduced by \cite{269,145,30} under the name Free Material Approach (FMA). There, the elastic properties at any point in the design space are decision variables. The method generates an optimal material distribution in terms of virtual homogenized material properties by the means of a direct search method and finite elements. The virtual material model prohibits a direct consideration of manufacturing constraints. In order to map the virtual elastic properties back to a real and manufacturable lay-up, some potentially expensive post-processing is required. A method to map the results of FMA to a tape-laying laminate plan is proposed by H"ornlein \textit{et al.} \cite{122}.

Voxel-based parameterization schemes covering the structure with a pre-defined set of virtual plies, each associated with an existing real material, a thickness and ply angle, and then searching for sets of voxels where these virtual plies should become existent or vanish have been proposed several times. The Discrete Material Optimization (DMO) approach has been pioneered by \cite{164,163,234,235}. It is inspired by SIMP methods and hence employs similar search algorithms. DMO addresses a laminate optimization problem. The method proposed by Hansel and Becker \cite{115} operates on a closely related parameterization scheme, but it is presented as a topology optimization method. In its initial implementation local optimality criteria guide an iterative search. In later approaches an \textit{EA} is employed instead (cf. \cite{114,116}).

A group of methods manually partitions the structure into geometric regions (or sections) in a first step. Then, in a second step a laminate parameterization scheme can be applied to the sections independently. In order to ensure the global cohesion of the structure, some plies covering and hence connecting several sections have to be applied \cite{5,226}. Furthermore, the optimization of the thickness of each section requires special attention: Whilst some methods fall back on (quasi-)isotropic models to determine thickness distributions (like \cite{207,212}), Giger and Ermanni \cite{90} encode the physical existence of plies by additional decision variables (Fig. 1.4).

Locally varying laminates can be achieved by accepting an additional attribute associated with each ply, i.e. the geometric region it covers on the structure. The concept of \textit{global plies} directly parameterizes this region: A
Figure 1.4: Lamination zones [90]: The structure is partitioned to geometric zones. Each zone is covered by a predefined number of plies. The attributes of these plies are collected in a heterogeneous parameter list. The existence of plies is encoded as a binary attribute allowing for variable thickness laminates.

Locally varying laminate is understood as the result of the application of fabric patches onto the structure. Their overlappings then lead to sections of different laminates in different geometric regions (Fig. 1.5). The concept of global plies in laminate optimization is introduced by [262, 263, 264]. There, the geometry attribute of each global ply is associated with a parameterized geometric sketch in a CAD-environment (Fig. 1.6). An EA then optimizes the patch shapes, materials, and orientations. By interpreting a patch as a parameterized component, structured, variable-length genotypes in an EA-framework may be employed (Chap. 5 of [89], [219], Fig. 1.7). An advantage of these parameterization schemes over DMO related approaches is their elegant way to encode variable local laminate thicknesses just by enlarging the search space by additional plies where appropriate.

The initial choice of the geometric shape of the sketches biases the direction of the search. In order to reintroduce more freedom in the design, the approach has been brought back to a voxel-based representation ([134, 93], revisited in Chap. 7 of [89]). A graph-based genotype maps the adjacency of the finite elements (serving as voxels here) in order to restrict the evolutionary search to connected patch shapes actually representable on a predefined finite element mesh of the structure (Fig. 1.8). Although generating promising results, the computational requirements prohibit the application of the approach in an industrial environment. Hence, an additional level of abstrac-
Section 4 consists of patch A and C, and section 4 is a laminate from A and C.

Figure 1.5: Mapping of three patches (A, B, and C) to four sections (1, 2, 3, and 4). Section 1 consists of patch A and B, section 2 of B and C, section 3 of only C, and section 4 is a laminate from A and C.
1.1 State-of-the-art

Figure 1.6: CAD-based patches [262]: Global plies are encoded by a heterogeneous parameter list. The shape of reinforcements is defined by geometric parameters of CAD-based sketches.

<table>
<thead>
<tr>
<th>Ply</th>
<th>Shape</th>
<th>Material</th>
<th>Angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x1, x2, x3,...</td>
<td>HTUD</td>
<td>90</td>
</tr>
<tr>
<td>2</td>
<td>y1, y2, y3,...</td>
<td>HM</td>
<td>45</td>
</tr>
</tbody>
</table>

Figure 1.7: Geometry-based patches [89, 219]: Primitive geometries (like rectangles or ellipses) are used to encode the shape of reinforcements. A heterogeneous, variable-length genotype allows for laminates with variable thickness.

<table>
<thead>
<tr>
<th>Ply</th>
<th>Shape</th>
<th>Material</th>
<th>Angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x, y, w, h</td>
<td>HTUD</td>
<td>90</td>
</tr>
<tr>
<td>2</td>
<td>...</td>
<td>HM</td>
<td>45</td>
</tr>
</tbody>
</table>
Figure 1.8: Graph-based patches [93]: The reinforcement shape is encoded as a subgraph of the element adjacency graph. A heterogeneous, variable-length genotype allows for laminates with variable thickness.

1.1.3 Combined topology, shape, and laminate optimization

Apart from the already introduced connections between topology and laminate optimization methods, there are some approaches covering laminate and shape optimization. This combination is attractive in the neighborhood of concentrated load introduction points or other stress concentrations like cutouts (e.g. [159, 28, 78]). Zu et al. [270] propose a combined shape and laminate optimization method for the optimization of pressure vessels manufactured by filament winding. Gamboni [84] investigates an extension of the CAD-methodology introduced by [262] with the possibility to consider additional shape parameters. de Boer [62] investigates aspects related to shape changes of fabric reinforced structures. Iuspa and Ruocco [126] investigate an optimization method for topological optimal cross-section geometries of stringer-stiffened panels.
1.2 Motivation and Goals

The work presented in this thesis founds on knowledge gained in \cite{262, 89, 134, 133} and summarized in Tab. 1.1:

A component-based parameterization of global plies as proposed by Zehnder \cite{262} is a key to the optimization of laminated composites. However, the approach of parameterizing the shape of each ply becomes computationally expensive when extending the scope from designing a few reinforcements towards an optimization of the entire laminate on a structure or when unbiased and hence more complex patch shapes shall be employed. Then, the number of parameters involved grows rapidly to an unaffordable level.

The approach presented by Giger \cite{89} to encode the laminate by a structured, variable-length genotype is able to reduce the parameter requirements to the bare minimum required by avoiding non-coding regions, i.e. parameters describing attributes of physically inexistent patches.

As a side effect of the graph-based parameterization presented by \cite{134, 93}, the formulation of the laminate optimization problem experienced a change: Instead of seeking boundaries of patches in a parametric domain, the method now aims at identifying subsets of predefined boundary components (here as edges of voxels). Thus, components of the boundaries may be shared among different patches. Hence, the shape attribute is partially decoupled

![Figure 1.9: Zonal graph-based patches \cite{133, 89}: The reinforcement shape is encoded as a subgraph of the adjacency graph of predefined zones (element sets). A heterogeneous, variable-length genotype allows for laminates with variable thickness.](image)
<table>
<thead>
<tr>
<th>Method</th>
<th>design freedom</th>
<th>parametric requirements</th>
<th>global plies</th>
<th>variable-length</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lamination zones [90], Fig. 1.4</td>
<td>low</td>
<td>low</td>
<td>no</td>
<td>no</td>
<td>• heterogeneous parameterlist with binary encoding of ply existence</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• connectivity has to be ensured by additional constraints or immutable layers</td>
</tr>
<tr>
<td>CAD-based patches [262], Fig. 1.6</td>
<td>medium</td>
<td>medium</td>
<td>yes</td>
<td>no</td>
<td>• heterogeneous parameterlist</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• patch shape by parameterized CAD-sketches</td>
</tr>
<tr>
<td>Geometry-based patches [89, 219], Fig. 1.7</td>
<td>medium</td>
<td>medium</td>
<td>yes</td>
<td>yes</td>
<td>• heterogeneous parameterlist of variable-length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• patch shape by primitive geometries like rectangles or ellipses</td>
</tr>
<tr>
<td>Graph-based patches [93], Fig. 1.8</td>
<td>high</td>
<td>high</td>
<td>yes</td>
<td>yes</td>
<td>• finite-element-based representation of patch shape</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• connectivity ensured by graph concepts</td>
</tr>
<tr>
<td>Zonal graph-based patches [133, 89], Fig. 1.9</td>
<td>medium</td>
<td>high</td>
<td>yes</td>
<td>yes</td>
<td>• lamination-zone-based representation of patch shape</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• connectivity ensured by graph concepts</td>
</tr>
</tbody>
</table>

Table 1.1: Overview of evolutionary laminate optimization methods with problem-specific representation.
from the single patch. However, the representation on the level of a finite element discretization had to be abandoned in order to achieve an affordable performance at the cost of a biased partition into predefined sections, [133].

The goals of this thesis are to investigate methods to overcome the above identified deficiencies of already proposed approaches without cutting their strengths. These are:

- the applicability to a wide variety of geometries, loads, and boundary conditions.

- Methodological independence from concrete objectives and constraints. This contrasts with local optimality methods outlined above.

- A reduced or ideally no post-processing step is required to obtain a human interpretable result from the optimization output.

These requirements form constraints for the following goals:

- A tailoring of the optimization algorithm to the traits of laminate optimization problems shall identify possibilities to reduce the numerical requirements and improve the reproducibility of results. Results presented in [89, 134, 93] are obtained in several ten to hundred-thousands of evaluations. Nonetheless, these results show a considerable spread in the solution quality in different runs on the same problem (e.g. Sec. 7.5 in [89]).

- The component-based genetic representation derived from the patch concept shall be extended by a method allowing for a unbiased optimization of laminate properties. A bias is introduced by predefined geometric sketches describing the patch shape. Hence, special attention and new conceptual approaches in the implementation of the patches shape attribute are required.

- The already introduced connection between laminate and topology optimization shall be exploited by consequently applying patches to parameterize a composite structure. This allows for variable-thickness and also void regions. The structures shape is then determined by the geometric boundaries of laminate sections with a concrete thickness.

- In order to remove the bias from the zonal approach ([133] and Chap. 7 in [89]), the partitioning of the structure into laminate sections shall be a result of the optimization and no longer user-defined input data.
Hence, the formulation of a laminate problem as the search for connected subsets of sections on a partitioned structure shall be augmented by allowing movable section boundaries.

1.3 Thesis outline

The structure of this thesis follows a gradual development of ideas and concepts. The chapters are organized in three parts. The first part presents and investigates generic methodologies in evolutionary structural optimization. The two following parts focus on particular aspects for the optimization and automated design of laminated composite structures.

The first part introduces algorithmic concepts employed throughout this thesis. Chap. 2 introduces a canonical EA serving as a foundation for further developments and investigations. It presents the relevant literature and introduces the terminology required to describe later adaptions. A special focus is set on representation concepts and thus the variation state. Chap. 3 develops a generalization of variable-length genetic crossover operators. The importance of variable-length representations in laminate and topology optimization is demonstrated. It presents an existing inter-species crossover introduced by [89]. This crossover is a source of illegal solutions if length constraints exist. Hence, length-constraints are incorporated in the variation operation. Chap. 4 introduces an asynchronous, parallel EA incorporating a deterministic local search disguised as a variation operator. This enables for the non-convex optimization in continuous search spaces. A niching method balances the stochastic and deterministic component of the algorithm. Asynchronism is used to allow for different runtimes of deterministic and stochastic variation operations.

The second part covers the optimization of laminate attributes. Chap. 5 investigates ply angle optimization problems. Numerical experiments and two application examples are used to demonstrate the properties of the algorithm developed in Chap. 4. Chap. 6 introduces a genetic encoding method for the evolutionary optimization of locally varying laminates incorporating local gradient information. It operates on structures with predefined geometric partitions. Global plies are used to allow for local reinforcements. A variable-length genotype and graph-based variation operations characterize the evolutionary search.

The third part investigates the encoding of topological attributes. Chap. 7 describes a method for the encoding of spline arrangements to optimize the shape and topology of monolithic isotropic structures. Again, a variable-length representation is employed. It encodes geometric attributes. The
1.3 Thesis outline

method is compared to a state-of-the-art SIMP approach. Chap. 8 discusses issues on the robustness of geometric variation operators introduced before. It illustrates possible reasons for the failure of geometric operations and automated meshing. It documents countermeasures and alternatives. Chap. 9 fuses the approach of Chap. 6 and Chap. 7 to achieve a combined topology and laminate optimization. The ability to optimize locally varying laminates is compared to the method of Chap. 6. Numerical experiments are used to investigate the combined optimization of topology, shape and laminate.

Chap. 10 collects the results. It draws final conclusions and gives recommendations for further research. Moreover, three concrete, continuative ideas emerging from this thesis project are outlined.

Two appendices contain further material: appendix A documents investigations on niching strategies leading to the algorithm finally presented in Chap. 4. Appendix B lists material models used in applications and numerical experiments.
Chapter 2

A canonical Evolutionary Algorithm

This chapter introduces foundations of EAs. It presents the relevant literature and introduces concepts, notation, and terminology used in the following chapters. The inspiration and deduced algorithmic sequence of a canonical EA is outlined. A particular focus is set on representation schemes. The fitness evaluation and corresponding constraint handling techniques are explained.

2.1 Terminology

2.1.1 A general optimization problem

A general, non-linear, constrained optimization problem can be brought to the form:

$$\text{find } \{x^*\} = \arg \min_{x \in \bar{X}} F(x)$$

$$= \{x^* \in \bar{X} : F(x^*) \preceq F(x) \forall x \in \bar{X}\}$$

subject to : $$\bar{X} = \{x \in X : g(x) \leq 0 \text{ and } h(x) = 0\}$$

$x^*$ is called a global optimal solution. The feasible region $\bar{X}$ is a subset of the search space $X$. It is determined by a set of inequality conditions $g(x)$ and equality conditions $h(x)$. $F(x) \in \mathbb{Y}$ is called objective. The symbol $\preceq$ denotes an ordering in the objective space $\mathbb{Y}$.
2.1.2 Classes of optimization algorithms

Numerical optimization methods can be discerned into two main classes: methods of Mathematical Programming (MP) or more generally called deterministic methods (cf. e.g. [186]) and stochastic optimization methods (cf. e.g. [232]). MP methods apply mathematical rules at each step in their iteration to determine the next iterate. These rules typically rely on assumptions regarding the optimization problem at hand. If these assumptions are not met, the method may fail or get stuck in a local optimal solution. Most numerical optimization algorithms of MP focus on problems with

- continuous, homogeneous search spaces $\mathbb{X}$ of fixed dimensionality (like $\mathbb{R}^n$),
- continuous, scalar objective spaces $\mathbb{Y}$ (like $\mathbb{R}$),
- convex, smooth objective functions $F(x)$ with steady derivatives up to order $k$ (where $k$ denotes the method order), and
- simply connected feasible regions $\bar{X}$ bound by smooth constraints $g(x)$ and $h(x)$ with steady derivatives up to order $k$.

Stochastic optimization algorithms are usually employed in fields where one or more of the above restrictions prohibit the application of MP methods. However, if MP methods are applicable these are prone to require less computational efforts than a stochastic optimizer would do. A stochastic optimization algorithm features one or both of the following properties:

Spall [232]:

1. There is random noise in the measurements of the criterion to be optimized and/or related information.
2. There is a random choice made in the search direction as the algorithm iterates toward a solution.

2.1.3 Evolutionary algorithms

EAs are stochastic, biologically-inspired, iterative optimization methods. Stochastic refers to the random contribution involved in the algorithms decision making process. Biologically-inspired accounts for certain algorithmic components mimicking mechanisms in organic evolution (cf. e.g. [64] for a survey of nature inspired algorithms).

The term EA is a more general term for four disciplines emerging in the field of evolutionary computation or artificial evolution (in historical order) [223]:
• Evolutionary Programming (EP), pioneered by Fogel et al. [81], aims at evolving artificial intelligence by population-based algorithms incorporating random variation operations and an elitist environmental selection stage.

• Evolution Strategies (ESs), introduced by Rechenberg [208] and Schwefel [225], apply variation and selection operations on a population in a loop. The original application is the optimization of parametric functions. As far as real-valued search spaces are considered, typical variation is Gaussian mutation, i.e. the addition of normally distributed values with zero mean and standard deviation $\sigma$, denoting the mutation strength. Research on ESs experienced a revival upon Baluja [21] establishing a link to artificial learning of probability distributions. Based thereon, a class of distribution estimating algorithms with a prominent member called Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [117] emerged.

• Genetic Algorithms (GAs) introduced by Holland [120] and later investigated by Goldberg [97], De Jong [66] and many others employ an encoding of solutions in a so called genotype space, which is then mapped to phenotype space before evaluation. A typical set of variation operators include mutation and crossover. Initially, GA is understood as an algorithm operating on a bit-string representation. In the meanwhile, the term is widened to comprise methods establishing their own representation concept associated with specialized variation operations.

• Genetic Programming (GP) founds on the work of Koza [146, 147]. It aims at evolving computer programs by employing evolutionary mechanisms on an alphabet of functional operators. The method has been proposed on genotypes representable as trees. Genetic operations are then able to exchange subtrees and mutate leafs. Extensions to other representations have been investigated (see e.g. [23]).

The field of evolutionary computation received much attention during the last four decades in research and application [151]. The high interest promoted contributions fusing and merging concepts over boundaries of the aforementioned four disciplines. Thus, a large number of popular methods in the field share features of more than one of the above categories. Hence, in this thesis the general term EA in the sense of Bentley [40] is preferred over a strict distinction of classes in evolutionary computation. In the following, the basic concepts in EAs are outlined. An overview of components in an EA is given in Sec. 2.2.
EAs are inspired by the mechanism driving organic evolution first proposed by Darwin [60] and usually referred to as the Darwinian principle of *survival of the fittest*. Notably, the phrase has been coined by Spencer [233] and was later adopted by Darwin as a synonym for his original term *natural selection*. In the context of evolutionary processes, *fit* is understood as the degree up to which an individual is adapted to its environment.

The biological inspiration founding EA-principles can be rediscovered in the terminology used in evolutionary computation which borrows a lot from genetics, evolutionary theory, and cellular biology: A candidate solution is called an *individual*. Its representation $x$ is called a genome or chromosome. A genome is constituted from single genes encoding attributes of an individual’s *phenotype*. The value of a gene is called allele. An individual is equipped with an attribute called *fitness* grading how well it competes in its environment. Depending on the implementation, the fitness has to be minimized or maximized. In MP the fitness usually corresponds to the objective $F(x)$. The current state of the search of an EA is not characterized by a single candidate solution but a set of individuals, i.e. a population. Correspondingly, the number of iterations are interpreted as generations of evolution.

Technical interpretations of organic evolution identified four principles characterizing the process: *reproduction*, *inheritance*, *variation*, and *selection* as stated in

Bentley [40]: 

..., as long as some individuals generate copies of themselves which inherit their parents’ characteristics with some small variation, and as long as some form of selection preferentially chooses some of the individuals to live and reproduce, evolution will occur.

Theoretical investigations were conducted to explain the common observations of an improved fitness in subsequent generations undergoing evolution as a collaborative work of the above principles. On the technical side the most important ones are the *Schema Theorem* [120], the *Building Blocks Hypothesis* [97], and several investigations on Markov chain translations (e.g. [222, 184, 156, 185]). These approaches either remain unproven theorems or hypotheses (like the first two), or focus on very special implementations or simplified aspects of EAs prohibiting them as a general theoretical foundation [210]. Hence, EAs are in general accepted as heuristic methods resulting in *educated guesses*. For prominent methods from MP theoretical proofs with respect to the solution quality and the algorithmic convergence can be given under their respective assumptions regarding the optimization problem. Nonetheless, these theoretical proofs are often of limited practical importance,
since there is often only little knowledge about the properties of the objective. This is also emphasized by Spall [232] in his explanation of the before quoted two properties of stochastic algorithms:

Spall [232]: The above two properties contrast with classical deterministic search and optimization, where it is assumed that one has perfect information about the loss function (and derivatives, if relevant) and that this information is used to determine the search direction in a deterministic manner at every step of the algorithm. In many practical problems, such information is not available, indicating that deterministic algorithms are inappropriate.

### 2.2 Components of an EA

EAs adopt the above four principles from biological evolution. In order to execute the process on a computer in limited time, these are extended by initialization and termination. The influence of an environment has to be simulated by an artificial evaluation function. The basic structure of an EA is outlined in Alg. 1. The algorithm operates on a population $M_{(t)}$ of individuals $x$. The population $M$ is updated in each iteration. The counter $t$ is used to count generations. The evolutionary cycle (State 4 to 9) models organic evolution. Thus, first a mating selection step (5) selects a subset $M'$ – the so-called genitors – from the parent population $M_{(t)}$. A variation step (6) then generates a new set of individual $M''$ – the offspring. These individuals need to be evaluated (7). Environmental selection updates the population.

---

**Algorithm 1** Evolutionary Algorithm Scheme

1: Initialize a population $M_0 \leftarrow \{i\cdot x\}$
2: Initialize the iteration counter $t \leftarrow 0$
3: Evaluate the population $M_0$
4: while continue($M_{(t)}, t) = 1$ do
5:  $M' \leftarrow$ mating selection from $M_{(t)}$
6:  $M'' \leftarrow$ variation from $M'$
7:  Evaluate the population $M''$
8:  $M_{(t+1)} \leftarrow$ environmental selection from $M_{(t)}$ and $M''$
9:  Increment $t \leftarrow t + 1$
10: end while
11: return best solution $x^*$ in $M_{(t)}$
This is achieved by creating a population $M_{(t+1)}$ from $M_{(t)}$ and $M''$. In the following, the basic components of the algorithm are detailed.

**Initialization:** The role of the initialization state (State 1 in Alg. 1) is to instantiate an initial population. Common practice is a random initialization, where the genetic information in each individual is set by chance in order to sample the entire search space as good as possible. Especially in highly constrained problems or when a local search is preferred over a global search, seeding of an initial population by prior knowledge is an alternative approach.

**Evaluation:** An evaluation function (State 7 in Alg. 1) assigns a fitness to each individual. Typically, the evaluation relies on individuals properties which are not directly evident from its genetic information. Thus, each individual has to be mapped before evaluation. In GAs mapping describes the transition from genotype space to phenotype space. A typical mapping function is e.g. the transformation of a binary encoded individual to the corresponding real-valued representation in a bit string GA. In structural optimization the update of a structural model with a new set of parameters encoded in an individual is called mapping. The fitness rates the individuals and is required for the selection strategies in the algorithm. Fitness concepts are introduced in Sec. 2.4.

**Mating selection:** Mating selection’s role is to choose genitors $M'$ for breeding as a subset of the parent population $M_{(t)}$ (State 5 in Alg. 1). A wide variety of selection schemes have been proposed (see e.g. [113, 95]), prominent ones are tournament selection (e.g. [222]), roulette wheel selection (e.g. [18]), or Stochastic Universal Sampling (SUS) [19]. Mating selection (apart from environmental selection) exerts a selection pressure on the individuals in the current population by excluding some potentially less fit ones from breeding. The population should be driven to fitter solutions. Thus, fit individuals are preferred over non-fit ones for mating. Hence, selection schemes operate either on the raw fitness or on fitness rankings. Rank-based selection schemes require only ranking information and are hence scaling invariant. Fitness-based selection schemes require information on the actual fitness values and are thus sensitive to scaling. The fitness is however not the only feature relevant for mating selection: the genetic diversity inside the population has to be maintained at a certain level in order to allow the algorithm to escape from local optima (cf. e.g. [156]). Hence, most selection schemes use random weighted choices instead of deterministic procedures in order to give less fit individuals a chance to mate.
Variation: The variation state (State 6 in Alg. 1) implements the biological principles of inheritance, variance, and reproduction. Variation generates offspring individuals from the before selected parents. Its concrete realization depends on the genetic representation. However, derived from GA-approaches, there are two main genetic operations: crossover and mutation. Crossover recombines the genetic information of two parents in two offspring individuals. Thus, it aims at exploiting the information already available in the population. Mutation generates one offspring from a single parent by introducing slight, random variations in its genetic information. Thus, it allows for an exploration of the search space by introducing new genetic information. Variation operators should enable exploration of the entire search space. This is closely related to the requirement for completeness of a genetic representation scheme as explained in the following (Sec. 2.3). Variation operators are either combined sequentially, i.e. first crossover and second mutation on a portion of the offspring or proportional with relative application rates until all genitors participated once in breeding. The relative application rates of the variation operators are used to tune the balance between explorative and exploitative search. An overview of variation operations in parametric evolutionary structural optimization can be found in Chap. 4 of [143].

Environmental selection: Environmental selection (State 8 in Alg. 1) simulates environmental pressure. Most often, the population size is kept constant, thus, for any individual inserted into the next population another one has to be removed. Hence, environmental selection decides which individuals in the new offspring and the parent population will survive. The same principles used for mating selection apply to environmental selection. However, often a deterministic selection strategy is employed here.

Termination: The role of the termination criterion (State 4 in Alg. 1) is to halt the algorithm. Since there is no applicable, sufficient criterion identifying a global optimal solution, typical criteria stop after a certain number of evaluations, after a certain number of generations, after a certain time, if a target fitness is achieved, or if the relative change of the best fitness over several generations falls below a threshold value.

2.3 Representation concepts

Particularly relevant to structural optimization in the field of evolutionary computation are the categories evolutionary design optimization and creative
evolutionary design. The first category focuses on optimizing an existing design by changing attributes of parameterized features therein. This approach is mainly related to shape optimization and sizing (Sec. 1.1). The second approach aims at evolving creative new designs requiring only limited prior knowledge. It is commonly associated to topology optimization. The distinction of these two approaches is intrinsic in product development and by far not limited to evolutionary computation or structural optimization:

Choi and Kim [57]: The design of a structural system has two categories: designing a new structure and improving the existing structure to perform better.

However, the boundaries in evolutionary computation are not straight nor clearly visible in all applications.

The representation concept describes the abstract engineering concept applied in a concrete structural optimization application. It is a recipe describing how to map encoded genetic values onto a structural model for evaluation. A parameterization, as first introduced in the optimization model of Sec. 1.1, is a special case of a representation where the genotype consists of an array-like structure containing an entry (i.e. a gene) for each decision variable. The alleles of these genes map to the structures attributes. More abstract representations use in general a multi-dimensional, variably typed data container or association structure holding all information required to describe a phenotype solution, i.e. its structural model. The information is partitioned and possibly structured in a form allowing the algorithm to change and rearrange it by a set of representation-specific genetic variation operators.

The representation concept is a key to extending the scope of structural optimization from tuning existing designs to model innovation and creativity [265, 24]. This is also illustrated by Kicinger et al. [138] who established a notable connection between representation concepts in structural optimization and the five levels of innovation postulated by the theory of inventive problem solving (romanized from Russian: Teoriya Resheniya Izobretatel’skich Zadatch) (TRIZ) [14]. The first level, corresponds to routine design problems solved by well-known methods. According to [138] this level can be achieved by an EA employing only selection operations. The second level, i.e. minor improvements to an existing system, can be realized by well-known parametric approaches, i.e. an EA searching in a parameterized search space of fixed dimensionality. The third level marks a fundamental improvement to an existing system by methods known outside of the domain. This requires the use of so-called island EAs, where the knowledge transfer by individual exchange between isolated subpopulations can be interpreted as innovation
injection. An innovation building on top of a novel concept or technology marks the fourth level. It can be reached by an EA not only evolving values of known attributes but the attributes themselves. The fifth level according to the taxonomy of TRIZ is characterized by a rare discovery or pioneering invention of essentially a new system. Kicinger et al. [138] claim that this level can be reached by so called generative representations.

Gen and Cheng [88] bring the requirements characterizing an ideal representation concept to a concise form consisting of five independent properties:

**Complete**: The translation from the genotype space $X$ to phenotype space $Y$ (the mapping $f : X \rightarrow Y$) should be complete, i.e. there is an encoding for each solution (surjective):

$$\forall y \in Y \exists x \in X : f(x) = y$$  \hspace{1cm} (2.4)

**Non-redundant**: The mapping $f : X \rightarrow Y$ – should be non-redundant (i.e. injective):

$$\forall x, x' \in X : x \neq x' \Rightarrow f(x) \neq f(x')$$  \hspace{1cm} (2.5)

A particularly critical mapping which should be avoided is one-to-$n$, where a single genotype maps to several phenotype solutions. This could occur in time-dependent or stochastic representations and requires additional procedures to determine the actual phenotype solution encoded in a certain genotype. Together with the requirement on completeness, the mapping has to be bijective.

**Legal**: A genetic modification of a genotype $x$ should lead to a legal solution. Illegal (not to be confused with infeasible in the context of constraint violations) is a solution which does not map to a phenotype for a given problem.

**Lamarckian property**: The meaning of alleles should not be context sensitive. This implies, that the interpretation of gene values is independent of each other. If this property is not satisfied, offspring of genetic recombination does not inherit parent traits.

**Causal / local / continuity**: Small variations of the genotype information induce small variations in phenotype space. If this property is not satisfied, offspring of genetic mutation does not inherit parent traits.
One can discriminate linear and non-linear representations in general. Linear representations organize genes in an array- or vector-like structure. Non-linear representations use more complex structures like multi-dimensional grids, trees, graphs, etc. Another distinction can be made between fixed- and variable-length representations: Fixed-length representations operate on a fixed number of genes whilst variable-length representations evolve the dimensionality of the search space by changing the number of genes involved. Variable-length representations are a special subclass of dynamic (in contrary to static) representations, i.e. they change the structure of the genotype over generations. Direct representations encode actual features of the design, indirect representations encode rules how to construct these features. In the context of structural optimization, all representations allowing for topological changes are called open-ended. Parameterizations prohibit topological changes. Open-ended representations are inspired by organic processes where evolution manipulates the genetic plans for complex objects rather than the objects themselves. The organisms are then built from the plans via a developmental process called morphogenesis.

2.4 Fitness evaluation

The algorithmic investigations and applications presented in the following focus on single-objective, real-valued, constrained, non-linear optimization problems of the form:

\[
\min_{x \in X} F(x) \quad F(x) \in \mathbb{R} \\
\text{subject to } g(x) \leq 0 \quad g(x) \in \mathbb{R}^j \\
h(x) = 0 \quad h(x) \in \mathbb{R}^k.
\] (2.6) (2.7) (2.8)

The problem is characterized by a scalar, real-valued objective \( F \) and two sets of constraints: \( j \) inequality constraints \( g \) and \( k \) equality constraints \( h \). Most optimization algorithms are suitable for unconstrained problems, hence, the above problem is transformed to an unconstrained optimization problem by a so-called constraint-handling technique.

Constraint-handling techniques for evolutionary optimization can be classified into five categories: penalty methods, representation-based approaches, repair algorithms, separation of objective and constraints, and hybrid methods (see [59] and references therein). Prior work by König [143] established a scaling and penalty scheme for engineering applications which has been applied in subsequent projects in evolutionary structural optimization [258, 262, 89, 218].
The method builds on a scalar pseudo-objective \( F \) assembled from weighted \((w_i)\) demand and constraint-functions \( D_i \):

\[
F(x) = \sum_i w_i D_i(x).
\] (2.9)

Three types of demands have been proposed: design objective (maps \( F \)), upper/lower limit constraint (maps \( g \)), and target constraint (maps \( h \)). The demand function \( F D_i \) scales the objective to a value in the real interval between zero and one:

\[
F D_i (x) = (a F (x) + b)^\alpha
\] (2.10)

\[
a = \frac{1 - \sqrt{0.1}}{F_{init} - F_{estim}}
\] (2.11)

\[
b = 1 - a F_{estim}
\] (2.12)

Scaling is achieved by the parameters \( F_{init} \) (upper bound) and \( F_{estim} \) (lower bound). A real valued parameter \( \alpha \) is used to control the steepness of the demand function and commonly set equal to five. The constraint-functions add normalized penalties for constraint violations. The penalties allow for marginal constraint violations. This is achieved by a smoothed step function for inequality constraints:

\[
g D_i (x) = \frac{1}{1 + e^{-\lambda (g(x) - \Delta)}}
\] (2.13)

\[
\lambda = \frac{1}{C_{tol}} \left( \ln \left( \frac{1}{0.01} - 1 \right) - \ln \left( \frac{1}{0.5} - 1 \right) \right)
\] (2.14)

\[
\Delta = \frac{1}{\lambda} \ln \left( \frac{1}{0.01} - 1 \right).
\] (2.15)

The parameter \( C_{tol} \) is used to control the steepness of the smoothed step in the infeasible region. A similar approach with a two-sided step is used for equality constraints:

\[
h D_i (x) = \begin{cases} 
0 & \text{if } |h(x)| < C_{adm}, \\
1 - e^{-\frac{|h(x) - C_{adm}|}{2\sigma^2}} & \text{otherwise}
\end{cases}
\] (2.16)

\[
\sigma^2 = \frac{C_{tol} - C_{adm}}{-2 \ln(0.5)}
\] (2.17)

A region of width \( C_{adm} \) around the constraint is not penalized. \( C_{tol} \) is used to tune the steepness of the penalty in the infeasible region. Unlike other penalty
functions, these demands do not increase unboundedly with increasing constraint violation but reach a plateau at $D_i = 1$. In TOD one may encounter large constraint violations during random initialization or if variation induces rigorous topological changes. Thus, a large portion of the population may be situated in the plateau-region of the penalty function. But, due to the almost uniform penalty there, selection is unable to distinguish between these individuals and induce pressure towards the feasible region. This drawback has been detected by Giger [89] and an adaptive scheme has been proposed to counteract it (see Sec. 2.5.2 in [89]). The adaption mechanism modifies $w_i$ and $C_{tol}$ based on the constraint violations observed in each generation. It basically emulates a penalty which increases unboundedly with increasing constraint violations. However, the approach undermines the original motivation for a smoothed step function by abandoning a constant scaling of all fitness contributions to a value between zero and one for an unbiased weighing. It turned out to require some tuning in order to avoid numerical instabilities when multiple constraints have to be considered [135].

Hence, two different approaches are employed throughout this project: A separation approach and – if a scalar objective is preferred – an exterior penalty method.

### 2.4.1 Separation method

The two selection states (Alg. 1, state 5 and 8) both require at least an order among the candidate solutions $x \in X$. Therefore, the here presented method implements the idea of superiority of feasible over infeasible solutions proposed by Powell and Skolnick [201] as a minimal ordering criterion.

A fitness $\mathcal{F}$ is introduced based on the original optimization problem $(F, g, h)$. The fitness is required to be completely ordered, i.e. the application of comparison operators $<, \leq, =, \geq, >$ returns unambiguous results for each pair in $X$. Therefore, the constraint violation of a candidate solution is computed as:

$$c(x) = \sum_i s_i \max(g_i(x), 0) + \sum_j s_j |h_j(x)|, \quad c(x) \geq 0 \quad (2.18)$$

The positive, scalar valued coefficients $s_i$ and $s_j$ are used to scale the different constraints. They are problem specific input data. Based on the constraint violation the following rules apply to find an order in two candidate solutions
1x and 2x:

\[
F(1x) < F(2x) \iff \begin{cases} 
F(1x) < F(2x) & \text{if } c(1x) = c(2x) \\
(1x) < (2x) & \text{otherwise}
\end{cases} 
\] (2.19)

\[
F(1x) = F(2x) \iff (c(1x) = c(2x)) \land (F(1x) = F(2x))
\] (2.20)

The remaining comparison operations can easily be derived as complements of the above equations. The resulting fitness \( F \) has to be minimized, i.e. an ascending order in \( F \) equals the order from the best to the worst solution in a population. \( F \) strictly prefers feasible solutions over infeasible ones.

### 2.4.2 Exterior penalty method

If a scalar fitness \( F \) is desired, an exterior penalty method is applied (e.g. [45]):

\[
F(x) = \frac{F(x)}{F_0} + p \left( \sum_i s_i \max (g_i(x), 0)^2 + \sum_j s_j h_j(x)^2 \right) 
\] (2.21)

This formulation requires the same constraint scaling parameters \( s_i \) and \( s_j \), but as well, a real-valued, positive penalty factor \( p \). Additionally, it may be convenient to scale the objective by \( F_0 \). Again, the resulting fitness \( F \) has to be minimized.

This penalty approach is characterized by the balance between the objective and the constraints. It does not strictly prefer feasible over infeasible solutions. Hence, constraint violations may be compensated by lower objective values. If for the original problem the optimal solution is on a constraint, the optimal solution for the transformed problem has to be expected in the infeasible region. Increasing the penalty factor \( p \) will increase the weight of the penalty term over the objective term and hence shift the optimal point towards the feasible region. This is only possible at the cost of steeper and thus numerically more challenging penalty functions.
Chapter 3

A Split-and-Splice Variable-Length Crossover

This chapter introduces a variable-length crossover incorporating constraints on offspring length. It demonstrates the importance of variable-length encodings in laminate and topology optimization. An existing crossover from literature is taken as a base.

3.1 Introduction

Variable-length denotes a representation’s ability to evolve the number of encoded features and not only the features itself. Variable-length genotypes have repeatedly and successfully been applied to structural optimization problems like TOD of trusses [204, 260], planar TOD [110, 111, 230, 91], compliant mechanism design [218], or laminate optimization [93].

A laminate optimization example: The importance of variable-length representations is demonstrated in the following simplified laminate optimization problem. The mechanical properties of a panel shall be optimized. The panel is to be built from laminae with all the same material and thickness. Each ply covers the complete structure. Thus, the orientation angles and the number of plies remain decision variables. Manufacturing considerations impose restrictions on a minimum and maximum number of plies. Laminates with less than six ($l_{\text{min}} = 6$) plies or more than 17 ($l_{\text{max}} = 17$) plies respectively are considered illegal. A fixed-length, heterogeneous representa-
A Split-and-Splice Variable-Length Crossover

tion scheme following the approach presented by Giger and Ermanni \[90\] is given as:

\[
x = \{\kappa_1, \phi_1, \kappa_2, \phi_2, \kappa_3, \phi_3, \ldots, \kappa_{17}, \phi_{17}\}
\]

\[
= \{\langle \kappa_i, \phi_i \rangle \}^{17}, \kappa_i \in \{0, 1\}, \phi_i \in \mathbb{R}.
\]

The representation has the length of the maximum number of plies, i.e. 17. This follows directly from the demand for a complete representation (cf. Sec. 2.3). The binary gene \(\kappa_i\) is used to encode the physical existence of the \(i^{th}\) ply. The real valued gene \(\phi_i\) represents its orientation angle. The requirement on a minimum number of plies is not incorporated in the representation and has to be handled as a constraint (i.e. \(\sum_{i=1}^{17} \kappa_i \geq 6\)). Notably, ply angles of physically inexistent plies may occupy a considerable portion of the genotype (so-called non-coding regions). Moreover, this type of encoding requires 34 genes to represent at most 17 plies. An alternative variable-length representation is:

\[
x = \{\phi_1, \phi_2, \phi_3, \ldots, \phi_n\}
\]

\[
= \{\langle \phi_i \rangle \}^n, \phi_i \in \mathbb{R}, n \in \mathbb{Z}, 6 \leq n \leq 17.
\]

This representation employs \(n\) genes to represent \(n\) plies. Information on physically inexistent plies is just omitted. Obviously, such a representation requires adaptations in the variation operators. Variation needs to be able to change the genotype length by introducing or removing genes. Length constraints may be incorporated directly in the genetic representation. A mutation operator changing the genome length randomly inserts or removes a gene in a parent individual. No more genes might be added to a solution of length \(l_{\text{max}}\), respectively removed from a solution with \(l_{\text{min}}\) genes. Thus, the incorporation of length constraints in such a mutation operator is trivial. A procedure to include length constraints in variable-length crossover operators is non-trivial and discussed in the following.

**A general, variable-length crossover problem:** In variable-length representations individuals of different length (i.e. different species) coexist in the population. Genetic variation operators have to be defined in a way that mating partners of different length generate legal offspring. This chapter investigates the following, crossover problem as an abstraction of the above example: Given two parent solutions \(p_1x = \{p_{x_1}, p_{x_2}, \ldots, p_{x_{l_1}}\}\), and \(p_2x = \{p_{x_1}, p_{x_2}, \ldots, p_{x_{l_2}}\}\), define an algorithm which generates two offspring
Figure 3.1: The inter-species crossover problem: The parents $p_1^x$ and $p_2^x$ shall be recombined to two offspring $o_1^x$ and $o_2^x$. Each gene is illustrated as a square box. The length of legal individuals is bound by $l_{\text{min}}$ and $l_{\text{max}}$. Illustrated is an example case with $l_1 = 10$ and $l_2 = 14$. The crossover points in the parents are depicted as triangles ($h_1 = 5$, $h_2 = 8$).
individuals $\mathbf{x}_1 = \{x_1^1, x_1^2, \ldots, x_1^o_1\}$, and $\mathbf{x}_2 = \{x_2^1, x_2^2, \ldots, x_2^o_2\}$. The offspring shall be constructed as a recombination of the genetic information in the parents (Fig. 3.1). A linear representation shall be employed, i.e. parents and offspring exist in a search space $X$, where any genotype can be encoded as an array-like structure. The number of genes in parents and offspring (i.e. $l_1, l_2, o_1, o_2$) are in general unequal.

As a solution to the above problem, a so called inter-species crossover has been proposed by Giger [89] (Sec. 5.2.2 therein). The operator is a generalization of common one-point crossovers. Instead of one single crossover-point for both solutions he proposes two crossover-points, i.e. one for each parent. Then, the operator recombines head and tail segments to two offspring. The crossover-points are set independently for both parents as uniform random integers within the length of the corresponding parent (see Eq. 5.1 in [89]).

Subsequent applications of variable-length encodings in [89, 218, 134] revealed a desire to have strict bounds on the actual length of a genotype. These hard constraints may either be induced as a direct requirement of the optimization problem at hand, as an indirect requirement of the mapping strategy employed to transfer genotype information to legal phenotypes, or by measures counteracting bloat [22]. Bloat is phenomenon observed in the subclass of redundant, variable-length representations. There, a tendency for increasing genotype lengths exists. Thus, an EA typically accumulates more information in a genotype than the bare minimum required. This effect is undesired, since long genotypes decrease the performance of the evolutionary search. Bloat is also relevant in structural optimization applications. Sauter [218] employs a variable-length encoding for component-based TOD of compliant structures. The components are formed by parameterized beams. Since the beam length is a parameter as well, the algorithm may chose to encode the same physical component by just a single, long beam or by a sequence of short, connected beams. Bloat induces a preference for the second approach. Notably, bloat is not an issue for the aforementioned laminate optimization problem. This, due to the fact that the representation is non-redundant.

If length is a criterion determining legality, the above outlined inter-species crossover is a source of illegal solutions. Although there is no documented application of inter-species crossover on a representation without length-constraints, no further efforts have been taken to incorporate these constraints in the procedure in order to fulfill the requirement of a legal representation (Sec. 2.3). Instead, illegal solutions are removed from the population in the mapping state. Thus, they reduce the performance of the evolutionary search. In order to eliminate the crossover-operation as a source of illegal solutions an extension to the existing procedure shall be developed.
3.2 Incorporating length-constraints

We assume a lower \( l_{\text{min}} \) and an upper limit \( l_{\text{max}} \) constraint on the length of a legal genotype representation \( x \). Hence, the above introduced crossover problem is accompanied by the following inequalities:

\[
\begin{align*}
    l_{\text{min}} & \leq l_1 \leq l_{\text{max}} \quad (3.5) \\
    l_{\text{min}} & \leq l_2 \leq l_{\text{max}} \quad (3.6) \\
    l_{\text{min}} & \leq o_1 \leq l_{\text{max}} \quad (3.7) \\
    l_{\text{min}} & \leq o_2 \leq l_{\text{max}} \quad (3.8)
\end{align*}
\]

Parent \( p_1 x \) shall be split such that a head of length \( h_1 \) and a tail of length \( t_1 \) results, \( h_2 \) and \( t_2 \) denote the same lengths for parent \( p_2 x \). Since the parents length is split into a head and tail length we can write:

\[
\begin{align*}
    h_1 + t_1 &= l_1 \quad (3.9) \\
    h_2 + t_2 &= l_2 \quad (3.10)
\end{align*}
\]

The head lengths can be interpreted as crossover-points for each parent. For legal crossovers we have to assert:

\[
\begin{align*}
    0 &\leq h_1 \leq l_1 \quad (3.11) \\
    0 &\leq h_2 \leq l_2 \quad (3.12)
\end{align*}
\]

By plugging \( o_1 = h_1 + t_2 \) into (3.7) and \( o_2 = h_2 + t_1 \) into (3.8) we obtain the following inequalities for the offspring lengths:

\[
\begin{align*}
    l_{\text{min}} &\leq h_1 + t_2 \leq l_{\text{max}} \quad (3.13) \\
    l_{\text{min}} &\leq h_2 + t_1 \leq l_{\text{max}} \quad (3.14)
\end{align*}
\]

From these we obtain:

\[
\begin{align*}
    l_{\text{min}} - l_2 &\leq h_1 - h_2 \leq l_{\text{max}} - l_2 \quad \text{by (3.13) and (3.10)} \quad (3.15) \\
    -l_{\text{max}} + l_1 &\leq h_1 - h_2 \leq -l_{\text{min}} + l_1 \quad \text{by (3.14) and (3.9)} \quad (3.16)
\end{align*}
\]

By adding (3.12) to (3.16) we get

\[
- l_{\text{max}} + l_1 + 0 \leq h_1 \leq -l_{\text{min}} + l_1 + l_2 \quad (3.17)
\]

and after combining with (3.11) and isolating \( h_1 \) we get a necessary condition for legal \( h_1 \):

\[
\max \{-l_{\text{max}} + l_1, 0\} \leq h_1 \leq \min \{-l_{\text{min}} + l_1 + l_2, l_1\} \quad (3.18)
\]
Since (3.5) and (3.6) hold, this simplifies to (3.11). A condition for \( h_2 \) can be obtained by introducing \( \Delta h = h_1 - h_2 \) and adding (3.15) and (3.16):

\[
\max \{ l_{\text{min}} - l_2, -l_{\text{max}} + l_1 \} \leq \Delta h \leq \min \{ l_{\text{max}} - l_2, -l_{\text{min}} + l_1 \} \quad (3.19)
\]

### 3.3 Split-and-splice crossover

From the above findings, we can now construct a generic algorithm to solve the variable-length crossover problem. As can be seen, the crossover-points cannot be set independently for each parent. Nonetheless, the crossover-point for one parent can be set arbitrarily. Since there are no further assumptions on the parents and especially their order in the above derivation, the algorithm is not sensitive to ordering of the parents. Hence, we chose without loss of generality \( h_1 \) for the first crossover-point. The actual lengths of the parents and the choice of \( h_1 \) then determine bounds for a legal crossover-point for the second parent. A corresponding algorithm is outlined in Alg. 2.

**Algorithm 2** Variable-length split-and-splice crossover

{
\begin{align*}
\text{Require:} & \quad \text{parent } p_1 \mathbf{x} = \{ p_1 x_1, \ldots, p_1 x_{l_1} \} \\
\text{Require:} & \quad \text{parent } p_2 \mathbf{x} = \{ p_2 x_1, \ldots, p_2 x_{l_2} \} \\
\text{Require:} & \quad \text{size bounds } \{ l_{\text{min}}, l_{\text{max}} \} \\
\text{Ensure:} & \quad \text{offspring } \{ o_1 \mathbf{x}, o_2 \mathbf{x} \} \\
1: & \quad h_1 \leftarrow \text{uniform random from } \mathbb{Z} \text{ subject to (3.11)} \\
2: & \quad \Delta h \leftarrow \text{uniform random from } \mathbb{Z} \text{ subject to (3.19)} \\
3: & \quad o_1 \mathbf{x} \leftarrow \{ p_1 x_1, \ldots, p_1 x_h, p_2 x_{h+1} - \Delta h + 1, \ldots, p_2 x_{l_2} \} \\
4: & \quad o_2 \mathbf{x} \leftarrow \{ p_2 x_1, \ldots, p_2 x_h - \Delta h, p_1 x_{h+1}, p_1 x_{h+2}, \ldots, p_1 x_{l_1} \}
\end{align*}

### 3.4 Concluding remarks

Illegal solutions reduce the efficiency of the evolutionary search. If genotype-length can determine legality, length constraints have to be considered in the design of genetic variation operators. Hence, a generic solution of a variable-length crossover problem with constraints on length has been developed. It extends an existing inter-species crossover by embedding knowledge on the length constraints in the variation procedure. Since there are no further assumptions on the exact type of the respective genes \( x_i \) in any participating
candidate solution $\mathbf{x}$, the algorithm operates equally on a wide variety of popular, linear representations like bit-strings, integer-, or real-valued arrays, and functional chains.
A Split-and-Splice Variable-Length Crossover
This chapter investigates an extension of the basic EA scheme introduced in Chap. 2. It features an embedded deterministic, local search by a first-order method from MP. Thus, the algorithm has to be classified as a hybrid or memetic EA (c.f. [105] and references therein). Moreover, a parallelization of the basic evolutionary cycle is used to reduce runtime on one hand, and to allow for a contemporary execution of stochastic and deterministic search in the same framework on the other hand. The algorithm is designed with an extension to more complex representations in mind. Hence, the concept of a genetic representation and a variation state shall be preserved.

4.1 Introduction

As outlined in Sec. 1.1.2, laminate optimization problems expose some challenging properties: If in some form the orientation of anisotropic reinforcements is considered for optimization, objective and constraint functions involving mechanical responses are in general multi-modal, i.e. they have several local optima. Nonetheless, objectives and constraints can still be differentiable
at acceptable computational cost. Such gradient information could be used to speed up convergence and to improve the solution quality of an optimization method. However, methods from MP employing gradient information risk to get stuck in multi-modal objectives.

4.1.1 Parallelization and asynchronism

Parallelism arises naturally in population-based search methods. Hence, parallelization of EA codes is one of the most popular approaches to reduce runtime in evolutionary optimization. The increasing availability of multi-processor computing systems has initiated investigations in the field of parallel EAs during the 1980s. Already Pettey and Leuze [198] cite thirteen related publications in their theoretical analysis of parallel genetic algorithms for the years 1981 to 1989. The more recent review of Alba and Tomassini [8] lists 135 references in the field.

Algorithm 3 Evolutionary Algorithm Scheme

1: Initialize a population $M_0 \leftarrow \{x\}$
2: Initialize the iteration counter $t \leftarrow 0$
3: Evaluate the population $M_0$
4: while continue($M(t), t) = 1$ do
5: \hspace{1em} $M' \leftarrow$ mating selection from $M(t)$
6: \hspace{1em} $M'' \leftarrow$ variation from $M'$
7: \hspace{1em} Evaluate the population $M''$
8: \hspace{1em} $M(t+1) \leftarrow$ environmental selection from $M(t)$ and $M''$
9: \hspace{1em} Increment $t \leftarrow t + 1$
10: end while
11: return best solution $x^*$ in $M(t)$
For the further discussion, the canonical EA scheme introduced as Alg. 1 (p. 27) is recapitulated here as Alg. 3. Different variants of parallel EAs have been deduced from this scheme. Two main classes of parallel EAs are distinguished \cite{11, 165, 44, 209} (Fig. 4.1): \textit{Panmictic} EAs keep the algorithmic structure and especially the global population of a canonical EA. The panmictic approach is also called \textit{global parallelization}. \textit{Structured} EAs split the population into several sub-populations (demes, islands, individuals). Isolated subalgorithms execute EA principles in parallel on each subpopulation. A migration policy exchanges individuals between the subalgorithms.

\textbf{Panmictic population:} Especially in engineering applications the function evaluation (state 7 in Alg. 3) involves numerical simulations and hence requires by far most of the computing time spent in a single optimization run. Hence, parallelization of the evaluation state may considerably reduce the overall runtime of an evolutionary optimization task. This is possible, since the different individuals in the offspring do usually not share any information and can thus be processed separately. The subalgorithm executing the evaluation (possibly on a different Central Processing Unit (CPU) or compute node) is called \textit{slave}. The execution of the slave algorithms is controlled by a central \textit{master} program. The \textit{master} also executes the not parallelized components of Alg. 3 and keeps control over the population. This kind of parallelization scheme is usually referred to as (single-population) master-slave paradigm (cf. \cite{46, 125, 11}, Fig. 4.2(a)). The master-slave paradigm does not influence the search of an EA. The speedup is achieved by a higher number of evaluations per unit time, the overall number of evaluations required stays the same as in sequential EAs. The master-slave approach gained some popularity in evolutionary structural optimization (see e.g. \cite{149, 148}). The EA-framework established in \cite{143} and later employed by \cite{89, 218, 262} also builds on a master-slave architecture. The number of slaves concurrently in operation is limited by one or more available resources, i.e. number of CPUs, licenses of subsidiary evaluation software, memory etc. An analysis on the efficiency of master-slave approaches is given by \cite{51}. Due to the integer characteristics of the population size \( n \) and the number of slaves \( s \) the computational load on each slave is only equal for \( n \mod s = 0 \) which is in general not the case. Hence, even on a homogeneous system the time required to evaluate a population of \( n \) individuals is dominated by the the slowest slave process, i.e. the one with the highest load. On the other hand some resources are idle from the time on when the fastest slave has finished its evaluation task until the slowest slave terminates. The actual evaluation time is proportional to the inverse of the number of slaves. But the time required for communica-
Figure 4.2: Algorithm structures for different parallel EAs
tion with each slave increases linearly with the number of slaves. Therefore, the overall speedup achievable is typically lower than linear in the number of slaves. Based on these observations, Cantu-Paz [51] deduce a critical and an optimum number of slaves based on the communication-to-computation time ratio.

**From generational synchronization to asynchronism:** The parallel subalgorithms in a master-slave parallel EA execute the same task (usually the evaluation state) concurrently. It processes generations one after another. Therefore, the master-slave approach is characterized by a so-called *generational synchronization*. Environmental selection (State 8 in Alg. 3) operates on the whole offspring. Thus, the master process is required to wait for the slowest slave before processing environmental selection. Hence, environmental selection acts as a *synchronization point*. The speedup lost in this synchronization point may be considerable in networked, heterogeneous environments [26]. Asynchronous (also called *generation-less*) EAs have been proposed to overcome this drawback. This is achieved by alternative implementations of the individual’s life-cycle dissolving a strict generational evaluation. The modeling of an individual’s life-cylce is a central idea in evolutionary optimization. An individual is created as offspring in a variation operator and extinct by environmental selection. In a synchronous EA, the life-cycle model is an essential component of the generation concept: the individual lives in its generation, it has once the possibility to participate in mating and possibly gets replaced by offspring in environmental selection (end of life). Due to the generation concept, mating partners in these algorithms are of the same age, i.e. they live in the same generation. Asynchronous algorithms have a mating pool with individuals of different age and thus eliminate the need for a generational synchronization [9]. Parallel subalgorithms execute each an evolutionary cycle of mating selection, variation, evaluation, and environmental selection. Not all participating subalgorithms execute the same state at the same time – hence, the term asynchronous. Asynchronism requires special adaptions of mating and environmental selection to allow the concurrent selection of mating partners and replacement of parents. Two commonly used implementations are detailed in the following paragraph.

**Structured populations:** Asynchronism is achieved by partially decoupling selection from breeding (i.e. variation and evaluation of the offspring) in a specific way. Typically, emphasis is set on continuous breeding in order to exploit the system capacity of a parallel environment and therewith achieve a high speedup. In order to allow for continuous breeding, one may accept
the same individual to participate more often in parent selection compared to a sequential EAs. Additionally, parents and their offspring are part of the same mating pool. These share a large portion of identical genetic information. Both effects lead to a considerably accelerated spread of successful alleles over the whole population for asynchronous EAs. Hence, the genetic diversity in the population decreases more rapidly when compared to synchronous EAs [10]. This affects the rate as well as the location of convergence of the algorithm in that it increases the risk of premature convergence. Therefore, most asynchronous algorithms use measures to artificially increase the population diversity. Thence, selection mechanisms are restricted to only a subset of all individuals currently alive. Parallel subalgorithms are each equipped with their own subpopulation, also called *demes*. The different subalgorithms are expected to evolve along different paths leading to an increased overall diversity. The approach also reduces the need for broadcast communications like in panmictic approaches. Depending on the size of the subpopulations two main categories can be distinguished: *distributed* (Fig. 4.2(b)) and *cellular* (Fig. 4.2(c)) parallel EAs. In distributed EAs employ population sizes larger than one whilst cellular EAs use a single individual in each subalgorithm. Each subalgorithm of a distributed EAs is a complete, isolated EA like Alg. 3. The choice of mating partners and accordingly the parent replacement is reduced to the respective subpopulation. The subalgorithms are connected by a migration scheme which regularly moves individuals from one deme to another. A popular migration structure is a circular array of demes [8] (Fig. 4.2(b)). Due to the fact that they only consider one individual at a time, the selection states in subalgorithms of cellular EAs need special attention. The mating pool is typically assembled from individuals on the deme currently under consideration and its direct neighbors. Neighborhood is deduced from the migration structure. A commonly used migration structure in cellular EAs is the so-called two dimensional diffusion grid [8] (Fig. 4.2(c)). Unlike panmictic approaches, structured EAs do not require a central manager having knowledge of all individuals currently alive.

Especially early investigations of asynchronous EAs – a popular one being ASPARAGOS [98, 99, 181] – found on the following characteristics: a parallelization bound to local resources (usually one execution process per CPU), no central manager having knowledge of all individuals currently in the pool, and a communication scheme allowing for information exchange between execution processes providing selection mechanisms in a certain *neighborhood*. Neighborhood is typically understood in a physical meaning, i.e. neighbored nodes on a grid computer. In engineering applications these characteristics induce certain drawbacks: The primitive, algorithm-internal scheduling policy
may fail at balancing the system load in heterogeneous networks or when the available resources change over time. This is typically the case in industrial applications, where other tasks compete for the same resources. Furthermore, the current state of a distributed algorithm is difficult to monitor because it misses a central storage or manager. This complicates interactive operations as well as recovery or restart scenarios.

### 4.1.2 Speedup through hybridization

Apart from reducing the overall runtime by utilizing the EA intrinsic parallelization, runtime can be reduced by a hybridization with deterministic local search strategies. Local search strategies from MP are well developed for real-valued search spaces and convex scalar objectives and can therefore help to speed up exploitative components of an EA and thus even improve the solution quality. Hybridization with local search leads to so-called *memetic* algorithms [203], these are an established field of research [105, 240, 266, 190]. An approach incorporating local search in the breeding state is first proposed by Burke and Smith [50]. Applications in the field of structural optimization are presented by Schmidt and Thierauf [220].

Hybrid approaches become especially attractive if some form of sensitivity information is available for the evaluation function: This may be given in the form of analytic or semi-analytic differentiation schemes (like direct or adjoint sensitivities in structural analysis, cf. [107, 248]) or by automatic differentiation of the evaluation code itself (e.g. [104]).

### 4.1.3 Motivation and goals

The here proposed hybrid, asynchronous EA shall cover the following aspects:

- Partitioning of an EA scheme into pieces of work which can be processed in parallel. An optimal partitioning in terms of runtime speedup should minimize the need for a synchronization between concurrent processes and at the same time allow for a full utilization of all available resources.

- A solution quality which is competitive to common synchronous algorithms for small to medium problem sizes (number of dimensions ten to hundred).

- A scheme which is expandable to more abstract genetic representations like variable-length encodings.

- A way to solve the scheduling problem induced by parallelization.
4.2 Method

4.2.1 A canonical non-linear optimization problem

This study focuses on real-valued, unconstrained optimization problems of the form:

$$\minimize F(x) \in \mathbb{R}, x \in \mathbb{R}^N$$  \hspace{1cm} (4.1)

$F(x)$ is in general a non-linear, non-convex, but smooth objective function. Further on, gradient information $\partial F(x)/\partial x$ is assumed to be available.

4.2.2 Algorithm architecture

Our parallel algorithm consists of a master process, a certain number of worker processes, and a scheduler process (see Fig. 4.3). In order to avoid confusion with a master-slave approach, the term worker is used instead of slave, indicating its different role. A slave executes a subalgorithm synchronized by the master process. The subalgorithms executed by workers are out of synchronization. A database serves as shared central storage. Note, that there is no assumption on whether these processes run on different or just one single physical machine. There is particularly no presupposition on the kind of
4.2 Method

hardware like e.g. shared or distributed memory or the number of CPUs. Furthermore, the number of worker processes may change over time.

The scheduler process distributes pieces of work to available resources. Due to the before introduced layer of abstraction between the algorithm’s architecture, the system’s architecture, and the available resources, the inherent scheduling problem of the parallel algorithm can be solved independently from the optimization problem – a feature which is necessary if the resources are not constant over time. Optimal scheduling is an active field of research by itself. On shared memory systems with only hardware limitations in action the scheduling task may be realized by utilizing the operating system’s internal scheduler accompanied by some threading framework, on grid computers or if more volatile restrictions like e.g. licenses have to be considered, there are so-called queuing systems providing these facilities. The exact realization of the scheduler process is beyond the scope of this study. For the further discussion an abstract interface called job \( (j) \) is introduced: A job describes an atomic piece of work handled by the scheduler. It consists of a functional part transforming input data to output data. A job undergoes the states ready (defined, but not yet known to the scheduler), queued (submitted to the scheduler but not yet assigned to available resources), running (assigned to resources and in execution on a worker), and complete (no longer in the scheduler queue) in this chronological order. A job’s state \( S(j) \) can be queried from the scheduler at any time.

A database models a collection of information, which can be concurrently read and written by all participating processes. It records all candidate solutions visited by the algorithm. A database manager observes and controls input and output operations on the database. Concurrent read and write access to the same single record is mutually synchronized by the database manager – i.e. one can assume a write lock for the remaining processes if one process acquires write access to a specified record. The last written state remains available for reading until a write transaction is complete. For the here proposed EA the requirement for a lock on the level of multiple records is avoided by design in order to prevent deadlocking. As can be seen in the following, a minimal implementation of the algorithm requires at least the genes \( x \), lifetime (as explained in Sec. 4.2.4), fitness values \( F(x) \), association to demes (as explained in Sec. 4.2.3), and associations to niches (as explained in Sec. 4.2.8) of individuals to be recorded in the database. However, the storage can be extended to comprise a complete image of the current and even previous states of the algorithm and all its components and settings for a later analysis, restarts or recovery.

The partitioning and adaption of an EA scheme to the master and worker
processes will be developed in the following sections.

4.2.3 Maintaining genetic diversity

Due to the weak bindings between the algorithm’s architecture and the hardware it runs on, the choice of the number of demes $d$ as well as the migration scheme is free. In the here presented study, a circular array of simply connected demes is investigated (Fig. 4.4). This, since it is one of the most simplest and still efficient structures investigated in literature [10] and particularly easy to implement. Potential migration paths allow one individual $i \mathbf{x}^*$ of deme $i$ to migrate to deme $i + 1$ (or from deme $d$ to deme 0 respectively) if it is fitter than the individual with highest fitness on deme $i + 1$ (deme 0), see Alg. 4. Notably, all individuals are stored in the same database. The association to a particular deme is stored along with its other properties. All individuals currently participating in mating on any deme form the pool. The pool size $n$ is an application specific parameter and has to be set by the user. It corresponds to the population size in panmictic approaches. The number of individuals on each deme are thus $n/d$. However, due to randomized initialization, breeding, and case dependent migration this number may show some fluctuations.

Algorithm 4 Migration

Require: demes $D$
Require: migration frequency $f \in \mathbb{N}$
Require: migration counter $m_k$
Ensure: updated migration counter $m_k$
1: $N \leftarrow$ count evaluations
2: if $N - m_k \geq f$ then
3: $m_k \leftarrow N$
4: for $D \in D$ do
5: $D' \leftarrow$ migration target of deme $D$ according to Fig. 4.4
6: $i \mathbf{x}^* \leftarrow$ select one migrant by mating selection from deme $D$
7: $F_{\text{min}} \leftarrow$ maximum fitness on deme $D'$
8: if $F(i \mathbf{x}^*) \leq F_{\text{min}}$ then
9: assign $i \mathbf{x}^*$ to deme $D'$
10: end if
11: end for
12: end if
4.2.4 Life-cycle modeling and asynchronous selection

Environmental selection plays a key role in the life-cycle model in conventional EAs (State 8 in Alg. 3). It exposes the newly created individuals to environmental pressure. Environmental selection acts as a synchronization point in parallel EAs. Thus, adapted selection procedures have to be employed to enable asynchronism.

The here proposed method builds on a direct implementation of a lifetime model. Therefore, each individual is assigned a lifetime $l \in \mathbb{R}$. The lifetime encodes an individual’s life state. Only individuals with $l > 0$ are alive and thus members of the mating pool. An individual’s lifetime is initialized to $l_0$ at its first insertion into the pool. Without loss of generality $l_0$ is set equal to one. Each time the individual participates in environmental selection its lifetime decreases, i.e. there is new offspring partially replacing it. Once its lifetime reaches zero it is completely replaced by offspring. Hence, the environmental selection procedure is transferred from the population to the individual level. A lifetime concept is already proposed by [38], but there it is employed to control the productivity in the algorithm and ranges from zero to a randomly initialized maximum lifespan for each individual. The amount to which the lifetime is decreased upon an application of environmental selection
Algorithm 5 Environmental Selection

**Require:** evaluated offspring individuals $M''$

**Ensure:** update of database with new evaluated offspring

1: update lifetime of all evaluated individuals with lifetime $l > 0$ in the database by Eq. (4.2)
2: assign lifetime $l_0 = 1$ to all individuals in $M''$
3: insert all individuals in $M''$ into the database

determines the pool size. A simple update rule on the lifetime is

$$l_{(t+1)} = l_{(t)} - \Delta l$$  \hspace{1cm} (4.2)

where $l_{(t)}$ is the individual’s lifetime before the update, $l_{(t+1)}$ after the update and $\Delta l$ is determined from the following steady state assumptions: In sequential EAs $n$ offspring replace $n$ parents in environmental selection. Offspring is generated by variation operators. The number of offspring generated by a specific variation operator is $n_{off,i}$. Mutation operators typically generate one individual $n_{off,mutation} = 1$, crossover generates two $n_{off,crossover} = 2$. In order to generate $n$ individuals, a specific variation operator $OP_i$ gets applied $a_i$ times:

$$\sum_{i=1}^{n_{OP}} a_i n_{off,i} = n.$$  \hspace{1cm} (4.3)

The application of variation operators is usually done by chance. Therefore, each variation operator $OP_i$ is associated to a relative application rate $r_i$. These rates are application specific parameters set by the user. The number of operator applications $a_i$ is then proportional to the rates $r_i$ with a proportionality factor $k$:

$$a_i = r_i k.$$  \hspace{1cm} (4.4)

By plugging (4.4) into (4.3) $k$ evaluates to

$$k = \frac{n}{\sum_{i=1}^{n_{OP}} r_i n_{off,i}}.$$  \hspace{1cm} (4.5)

As detailed in the following (Sec. 4.2.5) the application of variation operators is done in parallel on the workers. Each variation operator application is encapsulated in a breeder job which also applies environmental selection to
integrate the offspring into the pool. Thus, when generating \( n \) offspring by \( m \) breeder jobs, environmental selection gets applied \( m \) times, where

\[
m = \sum_{i=1}^{n_{OP}} a_i = k \sum_{i=1}^{n_{OP}} r_i = \frac{n \sum_{i=1}^{n_{OP}} r_i}{\sum_{i=1}^{n_{OP}} r_i n_{off,i}}. \tag{4.6}
\]

In order to keep the pool size constant at \( n \), all individuals generated in one time step have to be replaced by offspring during the following \( m \) applications of environmental selection. Hence, an individual’s lifetime has to be decreased from \( l_0 \) to zero in \( m \) steps. This can be achieved by setting

\[
\Delta l = \frac{l_0}{m} = \frac{l_0 \sum_{i=1}^{n_{OP}} r_i n_{off,i}}{n \sum_{i=1}^{n_{OP}} r_i}. \tag{4.7}
\]

For the case with normalized \( r_i \), i.e. \( \sum_{i=1}^{n_{OP}} r_i = 1 \), and \( l_0 = 1 \), this simplifies to a normalized, average number of offspring per variation operator application:

\[
\Delta l = \frac{\sum_{i=1}^{n_{OP}} r_i n_{off,i}}{n}. \tag{4.8}
\]

Due to the constant \( \Delta l \), rule (4.2) is easy to implement (Alg. 5): It requires one update statement on the database. Since it can be processed on the individual level, environmental selection requires no explicit locks or other synchronization primitives. However, the pool size is only approximately constant at \( n \): The application of variation operators is by design subject to random. Thus, each \( a_i \) is in fact an expectation value of a random variable. Dynamic effects occur in the time range from initialization until a pool size of \( n \) is reached as well as after changes in the number of available resources or changes in the number of running worker processes due to load adapting operations of the scheduler. These are not considered in the above analysis.

### 4.2.5 Initialization and asynchronous breeding

This section focuses on the subalgorithms executed on the workers. These tasks are formulated as jobs. The jobs are created by the master as discussed in the following (Sec. 4.2.6). The worker subalgorithms decentralize initialization (State 1 of Alg. 3) and breeding (States 6 and 7). This enables for parallel execution of all individual related states of Alg. 3.

Initialization and evaluation are encapsulated in an initializer job (see Alg. 6). This job creates a single individual on a remote worker and inserts it into the database.
Algorithm 6 Initializer Job

Require: a deme $D$
Ensure: creation and evaluation of new offspring

1: create a random individual $x$ with $x_i$ at random in search space
2: compute $F$ for $x$
3: assign $x$ to deme $D$
4: assign lifetime $l_0 = 1$ to $x$
5: insert $x$ into database

Breeding comprises the reproduction of offspring from parent individuals and the evaluation of this offspring. This procedure forms the core of a so-called breeder job (Alg. 7). Thus, it requires parents to be selected from the pool. In a structured population the mating pool is reduced to a specific subpopulation, i.e. the current deme. Therefore, formally a local copy of the pool has to be selected from the database to perform mating selection on a remote worker. Communication with the database can be reduced drastically if attributes of individuals are loaded lazily: If tournament selection shall be employed as mating selection, only a random sample with $T$ individuals ($T$ being the tournament size) has to be loaded to select one parent. Moreover, since the selection only requires knowledge on the fitness of the participating individuals, only their fitness attribute needs to be known for selection. Once $n_{par,i}$ mating partners are identified these exact genotypes can be reconstructed from the database. Each breeder job executes a single variation operator $OP_i$, i.e. a specific mutation or crossover operator. The choice of this operator is done in the master process when creating the breeder job. Thus, each breeder job generates $n_{off,i}$ offspring $M''$ depending on the actual operator $OP_i$. Offspring is evaluated on the remote worker and inserted into the pool by applying environmental selection.

4.2.6 Master process

The master process (Alg. 8) controls the required functional chain remaining from the original EA (Alg. 3): It manages initialization followed by a cycle of variation and the newly introduced migration. The cycle is stopped if a termination criterion is met. Termination criteria could indicate on a number of evaluations, on time, on the solution quality etc. The master process keeps book of the jobs it creates and submits during operation. In order for the scheduler not to run out of jobs and thus not fully exploiting the system capacity, the number of queued jobs is controlled and maintained at a certain
4.2 Method

Algorithm 7 Breeder Job

Require: a variation operator $OP_i$
Require: a deme $D$
Require: a scheduling interval $\Delta t$ (here $\Delta t = 1s$)
Ensure: creation and evaluation of new offspring

1: $M \leftarrow$ select all evaluated individuals with lifetime $l > 0$ from deme $D$
2: while $\|M\| < n_{par,i}$ do
3: wait for a scheduling interval $\Delta t$
4: $M \leftarrow$ select all evaluated individuals with lifetime $l > 0$ from deme $D$
5: end while
6: $M' \leftarrow$ mating selection from $M$
7: $M'' \leftarrow$ apply $OP$ on $M'$
8: compute $F$ for each $x_i$ in $M''$
9: assign each $x_i$ to deme $D$
10: update database by environmental selection (see Alg. 5)

level (parameterized by $j_{max}$) by the manager.

4.2.7 Hybridization by a deterministic variation operator

Given the above framework of an asynchronous EA, there is no longer a requirement for all operators to have the same runtime. This opens up the possibility to address a deterministic method from within the EA by disguising it as a genetic variation operator. The methodology is investigated on a Quasi-Newton method Symmetric Rank-1 Update (SR1). The exact deterministic algorithm is described as Alg. 6.2 in the textbook of Nocedal and Wright [186] (whereas the solution to the quadratic program (6.27) therein can be found by Alg. 7.2 in the same source). This algorithm is adapted to a facade of a mutation operator, i.e. an operator requiring one parent and creating one offspring (Alg. 9).

4.2.8 Exploiting niches

Apart from an approximation to a stationary point $\_o\!\!x$, Alg. 9 establishes a local approximation of the Hessian at $\_o\!\!x$, i.e. $B_k \approx \partial^2 F(\_o\!\!x) / \partial x^2$. If $B_k$ is positive definite, the offspring $\_o\!\!x$ is an approximation to a local minimizer of $F(x)$. Hence, the information in $B_k$ is valuable for the further optimization since it describes the curvature of $F(x)$ close to $\_o\!\!x$, i.e. it models a niche
Algorithm 8 Asynchronous Evolutionary Algorithm

Require: pool size \( n \in \mathbb{N} \)
Require: number of demes \( d \in \mathbb{N} \)
Require: migration frequency \( f \in \mathbb{N} \)
Require: maximum size of jobbuffer \( j_{max} \in \mathbb{N} \)
Require: termination criterion \( c_t \)
Require: variation operators \( \{OP_i\} \) and relative application rates \( r = \{r_i\} \)

1: initialize database
2: initialize migration counter \( m_k \leftarrow 0 \)
3: initialize demes \( D \leftarrow \{D_i\} \forall \in [0,d) \)
4: initialize job buffer \( J \leftarrow \{j_i\} \forall \in [0,n) \) with \( j_i \leftarrow \) initializer job (Alg. 6, input: random deme from \( D \))
5: while not \( c_t() \) do
6: submit all \( j \in J \) where \( S(j) = \) ready to scheduler
7: remove jobs from \( J \) where \( S(j) = \) complete
8: \( n_q \leftarrow \) count jobs \( j \in J \) where \( S(j) = \) queued
9: for \( k \in (n_q,j_{max}] \) do
10: \( OP_i \leftarrow \) random item from \( \{OP_i\} \) according to rate \( r_i \)
11: \( D \leftarrow \) random item from \( D \)
12: \( j \leftarrow \) breeder job (Alg. 7, input: \( OP_i \) and \( D \))
13: append \( j \) to \( J \)
14: end for
15: do migration (Alg. 4, input: \( D, f, m_k \))
16: adapt niche model (Alg. 10) for each niche in the database
17: end while
18: wait until the last job \( j \in J \) has state \( S(j) = \) complete
19: return best individual \( x^* \) in database
Algorithm 9 SR1-mutation

Require: parent $1\mathbf{x}$

Require: initial estimate of Hessian $B_0$ (here $B_0 = \Sigma^{-1}$, where $\Sigma$ is an estimated covariance of the alleles in the pool of the deme containing $1\mathbf{x}$)

Require: tolerance $\epsilon \ll 1$

Ensure: one offspring individual $o\mathbf{x}$

Ensure: an approximation to the Hessian $B_k \approx \partial^2 F(o\mathbf{x}) / \partial o\mathbf{x}^2$

1: initialize $\mathbf{x}_0 \leftarrow 1\mathbf{x}$
2: while $\|\partial F(\mathbf{x}_k) / \partial \mathbf{x}_k\| > \epsilon$ do
3: find a new individual $\mathbf{x}_k$ and update $B_k$ by one iteration of SR1
4: assign lifetime $l = 0$ to $\mathbf{x}_k$
5: insert $\mathbf{x}_k$ into database
6: end while
7: return last visited individual as offspring $o\mathbf{x} = \mathbf{x}_k$

centered at $o\mathbf{x}$. Since SR1-mutation is a deterministic, local search, further optimizations starting sufficiently close to a known niche center will result in the same offspring $o\mathbf{x}$ (up to a tolerance imposed by $\epsilon$). A local response model of the objective at $o\mathbf{x}$ comprising the shape of the niche is given from a second order Taylor approximation of the form

$$
\tilde{F}(\mathbf{x}) = F(o\mathbf{x}) + [\nabla F(o\mathbf{x})]^T (\mathbf{x} - o\mathbf{x}) + (\mathbf{x} - o\mathbf{x})^T \frac{1}{2} B_k (\mathbf{x} - o\mathbf{x}) \quad (4.9)
$$

where $\|\nabla F(o\mathbf{x})\| \leq \epsilon$ and hence the linear term becomes negligible. Each offspring of Alg. 9 is marked as local optimal solution. Therefore, a niche model is built upon termination of Alg. 9. It comprises the approximated Hessian $B_k$ and the an association to the center $o\mathbf{x}$. Fitness values inside the niche may be approximated by (4.9). Obviously, this model is only appropriate in a region sufficiently close to $o\mathbf{x}$. An initial estimate for the size of the niche is assumed from the contour surface of $\tilde{F}(\mathbf{x})$ crossing the start point $1\mathbf{x}$ of the iterative path of Alg. 9. The set of contour surfaces of $\tilde{F}(\mathbf{x})$ can be parameterized by their quadratic term: Obviously, points $\mathbf{x}$ with equal values for the quadratic term are on the same contour surface, i.e. have the same $\tilde{F}$. Hence, the quadratic term interpreted as a scalar distance from the niche center $o\mathbf{x}$ to a point $\mathbf{x}$ on a contour surface $C$ is used to parameterize $C$:

$$
d(\mathbf{x}, o\mathbf{x}) = \sqrt{(\mathbf{x} - o\mathbf{x})^T \frac{1}{2} B_k (\mathbf{x} - o\mathbf{x})} \quad (4.10)
$$

d(\mathbf{x}, o\mathbf{x})$ is a form of the Mahalanobis distance [168]. This distance metric is attractive: A single scalar value can be used to characterize an ellipsoidal
boundary of a niche model following its spatial orientation and elongation. Thus, the ellipsoidal surface \( C \) identified by \( d_0 = d(1x, o\cdot x) \) (further referred to as *niche radius*) is used as a bound for a *niche* exploited by one application of Alg. 9. Now, for further SR1-mutations, the algorithm is stopped if an iterate \( x_k \) is inside an already known niche, i.e. \( d(x_k, o\cdot x) \leq d_0 \).

The approximation \( B_k \approx \partial^2 F(o\cdot x) / \partial o\cdot x^2 \) may be crude and the initial guess of the niche radius \( d_0 \) may not be accurate enough to bound one sink in the objective landscape. Thus, the stochastic component of the EA is not kept from evaluating more solutions inside a known niche. But, information gathered from these evaluations is used to adapt the local model describing the niche, i.e. to increase or decrease the niche radius. A niche adaption algorithm is outlined in Alg. 10. The adaption is controlled by a correlation coefficient \( \rho \) between evaluated values of \( F \) and predicted values of \( \tilde{F} \). The model (4.9) involves \( p = (N^2 + N)/2 \) coefficients in \( B_k \). Hence, the adaption is only carried out, if at least \( p \) evaluations inside the niche are available. Niche radius adaption is carried out for each known niche within the while-loop of Alg. 8.

**Algorithm 10** Niche radius adaption

**Require:** a niche \((o\cdot x, B_k, \tilde{F}, d_0)\)

**Require:** a list of evaluated individuals inside the niche \( X = [x_1, \ldots, x_p] \)

**Require:** adaption bounds \( r_{min}, r_{max} \in (0, 1) \) (here \( r_{min} = 0.9, r_{max} = 0.98 \))

**Require:** adaption parameter \( \alpha \in (0, 1) \) (here \( \alpha = 0.4 \))

**Ensure:** adapted niche radius \( d'_0 \)

1: if \( p \geq (N^2 + N)/2 \) then
2: \( f \leftarrow \{ F(x) \} \forall x \in X \)
3: \( \tilde{f} \leftarrow \{ \tilde{F}(x) \} \forall x \in X \)
4: \( \rho \leftarrow \) Spearman’s rank correlation between \( f \) and \( \tilde{f} \)
5: if \( \rho < r_{min} \) then
6: \( d'_0 \leftarrow d_0(1 - \alpha) \)
7: else if \( \rho > r_{max} \) then
8: \( d'_0 \leftarrow d_0(1 + \alpha) \)
9: end if
10: else
11: \( d'_0 \leftarrow d_0 \)
12: end if
13: return \( d'_0 \)
4.2.9 Niche-based fitness sharing

The niches established in Sec. 4.2.8 may not only be engaged to control further local searches. It is rather desirable to focus the search on unexploited regions and thus avoiding evaluations in already exploited niches. This is achieved by penalizing individuals inside niches. Hence, they are less attractive for mating selection and thus for further evolution. A commonly employed locality based penalty scheme is the fitness sharing concept first proposed by [120] and later elaborated by [96]. There, individuals in a certain niche get a penalty on their fitness. The penalty is determined by the number of individuals occupying the same niche, the so-called niche count. The name fitness sharing deduces from the idea to share the fitness among all individuals in the same niche. For the initially investigated maximization problems the conception of fitness sharing is directly evident as a division of the raw fitness by the niche count. A crucial component of niche sharing approaches is the determination of the position and size of niches. The here embedded local search solves this task reliably.

In order to use the available niche information for fitness sharing, the original concept is transformed from maximization problems to the here considered minimization problems. Furthermore, we extend the concept for a fitness which can take negative as well as positive values (cf. Appendix A). Hence, the shared fitness used in mating selection and migration is adjusted to:

\[ F' (x) = F (x) \cdot (1 + m_c (x))^{\text{sgn} F(x)}. \]  

(4.11)

The raw fitness \( F \) is increased for individuals inside niches. The amount of the increase depends on the niche count \( m_c (x) \). The niche count is the portion of the penalty to be taken by a specific individual \( x \). This portion depends on the proximity of an individual to all known niche centers \( i x \). It is computed from the sharing function \( \text{sh} \) over all niches \( i \) with niche centers \( i x \):

\[ m_c (x) = \sum_i \text{sh} (x, i x). \]  

(4.12)

The sharing function describes a normalized proximity of \( x \) to a niche with center \( i x \) and radius \( d_0 \):

\[ \text{sh} (x, i x) = \begin{cases} 1 - \frac{d(x, i x)}{d_0} & d(x, i x) \leq d_0 \\ 0 & \text{otherwise} \end{cases} \]  

(4.13)

Hence, the niche count \( m_c (x) \) is zero if \( x \) is not contained in any niche. Then, the individual’s fitness is not affected by sharing. Otherwise, the niche count...
is larger than zero and the fitness is increased. For simplicity the distances $d(x,i\vec{x})$ (Eq. 4.10) and predicted responses (Eq. 4.9) are computed and stored immediately after the evaluation of the individual. Accordingly, in the niche count $m_c(x)$ only niches known at the time of the individuals evaluation are considered. This strategy seems valid as long as the application rate of SR1-mutation is small compared to the other operators and hence the number of niches changes slowly compared to the number of evaluations over time.

4.3 Implementation

As a proof of concept the proposed algorithm is implemented in an object-oriented code in the Python programming language$^1$ (version 2.5). Grid Engine$^2$ is employed as a queuing system on a heterogeneous cluster consisting of 28 blade computers and two workstations with two CPUs each. MySQL$^3$ serves as a database server, InnoDB$^4$ is used as underlaying storage engine. The database service as well as the scheduler run on the same additional server computer.

The implementation is tailored to the investigation of structural optimization problems in mechanical engineering where the evaluation of a single individual typically requires several minutes. There, Grid Engine’s minimal scheduling interval of one second as well as the latency implied by an implementation in an interpreted language show only negligible influence on the overall execution time. In the following investigation of analytic objective functions which evaluate in splits of seconds the overall computation time is mainly dominated by the overhead of the realization on approval.

4.4 Numerical experiments

Numerical experiments shall be used to investigate the solution quality of the here proposed asynchronous EA. Therefore, five different configurations are compared:

SSYNCEA : sequential, synchronous EA

PASEA (d=1) : parallel, asynchronous EA with a panmictic pool (one deme)

---

$^1$http://www.python.org
$^2$http://gridengine.sunsource.net
$^3$http://www.mysql.com
$^4$http://www.innodb.com
Table 4.1: Test functions used in numerical experiments

<table>
<thead>
<tr>
<th>Name</th>
<th>Objective</th>
<th>Search space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sphere</td>
<td>( f(x) = \sum_{i=0}^{N-1} x_i^2 )</td>
<td>(-5.12 \leq x_i \leq 5.12)</td>
</tr>
<tr>
<td>Rosenbrock</td>
<td>( f(x) = \sum_{i=0}^{N-2} \left[ 100 \left( x_i^2 - x_{i+1} \right)^2 + (x_i - 1)^2 \right] )</td>
<td>(-10 \leq x_i \leq 10)</td>
</tr>
<tr>
<td>Michalewicz</td>
<td>( f(x) = -\sum_{i=0}^{N-1} \sin x_i \left( \sin \left( \frac{(i+1)}{\pi x_j} \right) \right)^{20} )</td>
<td>(0 \leq x_i \leq \pi)</td>
</tr>
<tr>
<td>Schwefel</td>
<td>( f(x) = 418.0826N - \sum_{i=0}^{N-1} \left( x_i \sin \sqrt{</td>
<td>x_i</td>
</tr>
<tr>
<td>Rastrigin</td>
<td>( f(x) = 10N + \sum_{i=0}^{N-1} \left( x_i^2 - 10 \cos \left( 2\pi x_j \right) \right) )</td>
<td>(-5.12 \leq x_i \leq 5.12)</td>
</tr>
</tbody>
</table>

PASEA (d=5) : parallel, asynchronous EA with five demes

PASEA (d=10) : parallel, asynchronous EA with ten demes

HYPASEA (d=5) : hybrid, parallel, asynchronous EA with five demes

All algorithms are analyzed on a testbed of established, real-encoded, both convex and non-convex benchmark problems (see Tab. 4.1, with the number of dimension set to \( N = 10 \)). All configurations use random initialization, Gaussian-Mutation, and Arithmetic-Crossover. The hybrid setup additionally employs SR1-mutation. The rates are set to 0.05 for Gaussian-Mutation and 0.95 for crossover in the non-hybrid configurations, respectively 0.05 for Gaussian-Mutation, 0.9 for crossover, 0.05 for SR1-mutation in the hybrid setup. Tournament selection is employed as mating selection (tournament size \( T = 3 \)). In the synchronous setup comma-selection is used as environmental selection and the population size \( n \) is set to 100. Migration frequency \( f \) is set to 500 for the asynchronous algorithms. All algorithms are stopped after 10000 function evaluations.

For each configuration and each benchmark function a set of ten independent runs is analyzed. This results in 250 EA runs with an overall of 2.5 millions of function evaluations. The solution quality is rated by the best solution so far, i.e. the minimal objective value found after \( k \) evaluations for a certain run. This gives a set of ten monotonically falling curves for each configuration and each function. These curves are averaged over all ten runs and plotted in subfigure (a) of Fig. 4.5 to Fig. 4.9 for the different benchmark functions. Tab. 4.2 summarizes the objectives values for the different functions and configurations. In order to get statistically founded data, the best fitness values of a configurations \( C_1 \) are compared to the best fitness values found by the reference configuration SSYNCEA. The comparison is based on a two-sided Wilcoxon-Rank-Sum-Test. The set of best fitness values
found is taken as test samples: For a certain \( k \) ten values of the reference configuration SSYNCEA and ten values of an alternative configuration \( C_1 \) are compared. PASEA (\( d=1 \)), PASEA (\( d=5 \)), PASEA (\( d=10 \)), and HYPASEA (\( d=5 \)) are each considered as alternative configurations \( C_1 \). The hypothesis \( H_0 (\text{SSYNCEA}=C_1) \) assumes that both samples come from the same distribution. The Rank-Sum-Test indicates on which probability (\( p \)-value) this hypothesis can be accepted. Here, a significance level of 5\% is taken, i.e. \( p \) values lower than 0.05 indicate that hypothesis \( H_0 \) is rejected, i.e. a significant difference in the two test samples. \( p \)-values are evaluated for each number of evaluations \( k \) from one to 10000 and for all alternative configuration \( C_1 \). The values are given in subfigure (b) of Fig. 4.5 to Fig. 4.9 for all benchmark functions. The \( p \)-values for the best fitness after 10000 evaluations are given in the last column of Tab. 4.2.

4.5 Results and Discussion

The non-hybrid, parallel, asynchronous configurations (PASEA (\( d=1 \)), PASEA (\( d=5 \)), PASEA (\( d=10 \))) find significantly poorer solutions when compared to the sequential, synchronous setup for four out of five investigated problems (Fig. 4.5, 4.6, 4.7, 4.9). A significant improvement can only be seen in the Schwefel test function (Fig. 4.8) for configuration PASEA (\( d=10 \)). An increasing number of demes from PASEA (\( d=1 \)), PASEA (\( d=5 \)), to PASEA (\( d=10 \)) helps to improve the solution quality in convex (Fig. 4.5 and Fig. 4.6) as well as non-convex functions. However, this tendency reaches significance only for the above mentioned case of PASEA (\( d=10 \)) on the non-convex Schwefel test function. A panmictic pool (PASEA (\( d=1 \))) increases the speed of convergence in the very beginning of the optimization which becomes significant for all but the Schwefel test function. All observations so far support the thesis outlined in Sec. 4.2.3, i.e. an increased loss of genetic diversity for the asynchronous configurations which depends on the structure of the population. Note, that the non-hybrid configurations cannot make use of the fitness sharing mechanism developed in Sec. 4.2.9 to counteract this loss since there is no niche-detection without SR1-mutation.

The hybrid configuration HYPASEA (\( d=5 \)) finds significantly better solutions than SSYNCEA for the convex functions Sphere and Rosenbrock (Fig. 4.5 and 4.6) and as well for the multi-modal Schwefel test function (Fig. 4.8). For the Michalewicz function (Fig. 4.7) significantly poorer solutions are found for \( k \) in a range from 1500 to about 8200 number of evaluations whereas significance is not reached from 8200 to 10000 evaluations. Only for the highly multi-modal Rastrigin function the hybrid approach significantly
4.5 Results and Discussion

<table>
<thead>
<tr>
<th>Function</th>
<th>Configuration</th>
<th>avg ($F^*$)</th>
<th>std ($F^*$)</th>
<th>p-values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sphere</td>
<td>SSYNCEA</td>
<td>5.6175 · 10^{-2}</td>
<td>5.2565 · 10^{-2}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PASEA (d=1)</td>
<td>3.2333 · 10^{-1}</td>
<td>1.0755 · 10^{-1}</td>
<td>4.3300 · 10^{-5}</td>
</tr>
<tr>
<td></td>
<td>PASEA (d=5)</td>
<td>3.0487 · 10^{-1}</td>
<td>2.1147 · 10^{-1}</td>
<td>6.8415 · 10^{-3}</td>
</tr>
<tr>
<td></td>
<td>PASEA (d=10)</td>
<td>3.4772 · 10^{-1}</td>
<td>1.5513 · 10^{-1}</td>
<td>7.2528 · 10^{-4}</td>
</tr>
<tr>
<td></td>
<td>HYPASEA (d=5)</td>
<td>1.9895 · 10^{-1}</td>
<td>1.0570 · 10^{-1}</td>
<td>1.0825 · 10^{-5}</td>
</tr>
<tr>
<td>Rosenbrock</td>
<td>SSYNCEA</td>
<td>5.4019 · 10^1</td>
<td>2.9995 · 10^1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PASEA (d=1)</td>
<td>8.1848 · 10^2</td>
<td>1.9169 · 10^3</td>
<td>3.2475 · 10^{-4}</td>
</tr>
<tr>
<td></td>
<td>PASEA (d=5)</td>
<td>6.5383 · 10^2</td>
<td>1.5740 · 10^3</td>
<td>2.0892 · 10^{-3}</td>
</tr>
<tr>
<td></td>
<td>PASEA (d=10)</td>
<td>3.6446 · 10^2</td>
<td>2.5318 · 10^2</td>
<td>7.5776 · 10^{-5}</td>
</tr>
<tr>
<td></td>
<td>HYPASEA (d=5)</td>
<td>9.2523 · 10^0</td>
<td>1.3785 · 10^0</td>
<td>1.0825 · 10^{-5}</td>
</tr>
<tr>
<td>Michalewicz</td>
<td>SSYNCEA</td>
<td>−7.1734 · 10^1</td>
<td>5.5925 · 10^{-1}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PASEA (d=1)</td>
<td>−6.8178 · 10^1</td>
<td>6.4520 · 10^{-1}</td>
<td>4.3587 · 10^{-1}</td>
</tr>
<tr>
<td></td>
<td>PASEA (d=5)</td>
<td>−5.2445 · 10^1</td>
<td>6.3992 · 10^{-1}</td>
<td>2.1650 · 10^{-5}</td>
</tr>
<tr>
<td></td>
<td>PASEA (d=10)</td>
<td>−4.6534 · 10^1</td>
<td>9.0352 · 10^{-1}</td>
<td>1.1496 · 10^{-2}</td>
</tr>
<tr>
<td></td>
<td>HYPASEA (d=5)</td>
<td>−5.9523 · 10^1</td>
<td>9.2020 · 10^{-1}</td>
<td>1.6549 · 10^{-1}</td>
</tr>
<tr>
<td>Schwefel</td>
<td>SSYNCEA</td>
<td>3.0043 · 10^3</td>
<td>4.2187 · 10^2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PASEA (d=1)</td>
<td>2.9783 · 10^3</td>
<td>3.7709 · 10^2</td>
<td>1.2301 · 10^{-1}</td>
</tr>
<tr>
<td></td>
<td>PASEA (d=5)</td>
<td>2.6392 · 10^3</td>
<td>3.5020 · 10^2</td>
<td>1.0512 · 10^{-1}</td>
</tr>
<tr>
<td></td>
<td>PASEA (d=10)</td>
<td>2.6937 · 10^3</td>
<td>4.9296 · 10^2</td>
<td>3.2475 · 10^{-4}</td>
</tr>
<tr>
<td></td>
<td>HYPASEA (d=5)</td>
<td>2.6146 · 10^3</td>
<td>5.4880 · 10^2</td>
<td>1.0825 · 10^{-5}</td>
</tr>
<tr>
<td>Rastrigin</td>
<td>SSYNCEA</td>
<td>1.5156 · 10^1</td>
<td>4.4490 · 10^0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PASEA (d=1)</td>
<td>2.5489 · 10^1</td>
<td>1.2220 · 10^0</td>
<td>1.8543 · 10^{-2}</td>
</tr>
<tr>
<td></td>
<td>PASEA (d=5)</td>
<td>5.3491 · 10^1</td>
<td>3.5020 · 10^1</td>
<td>1.0825 · 10^{-5}</td>
</tr>
<tr>
<td></td>
<td>PASEA (d=10)</td>
<td>6.6602 · 10^1</td>
<td>2.5921 · 10^1</td>
<td>1.0825 · 10^{-5}</td>
</tr>
<tr>
<td></td>
<td>HYPASEA (d=5)</td>
<td>5.0084 · 10^1</td>
<td>1.3929 · 10^1</td>
<td>1.0825 · 10^{-5}</td>
</tr>
</tbody>
</table>

Table 4.2: Lowest fitness value for each function and each configuration (average over all runs avg ($F^*$) and standard deviation over all runs std ($F^*$)) and p-values of Wilcoxon-Rank-Sum tests for the equality hypotheses in comparison to configuration SSYNCEA after 10000 evaluations.
Figure 4.5: Results for the Sphere test function
4.5 Results and Discussion

![Graph showing results for the Rosenbrock test function](image)

**Figure 4.6:** Results for the Rosenbrock test function
Figure 4.7: Results for the Michalewicz test function
Figure 4.8: Results for the Schwefel test function
Figure 4.9: Results for the Rastrigin test function
reduces solution quality when compared to SSYNCEA. This observation may be explained by the SR1-mutation getting repeatedly stuck in local optimal solutions. Accordingly, it requires many evaluations to generate only slight improvements. Each local niche model covers only a very small part of the search space there. Thus, it has only a small impact on the further search but requires many function evaluations during its creation.

4.6 Concluding remarks

An asynchronous, parallel evolutionary algorithm scheme is presented with the following traits:

- It has an abstract architecture with essentially no bindings to hardware.

- It separates the optimization problem tackled by the EA mechanisms from the scheduling problem induced by parallelization. Thus, an algorithm external scheduler may be employed to solve the scheduling problem. Hence, depending on the scheduler the algorithm works not only on grid computers but as well on heterogeneous systems with time variable resources.

- The partitioning of the algorithm into pieces of work (jobs) founds on an analysis of its components. The conceptual changes from a synchronous EA are mainly limited to the environmental selection state. There, a procedure designed to transfer the selection operation from the level of populations to the individual level is proposed. This method eliminates the requirement for population-based synchronization primitives.

- A central storage in the form of a database is employed to record the current state of the evolutionary search. This allows for monitoring, restarting, or recovering the state of an evolutionary search in practical applications.

- Since variation operates in parallel, a deterministic local search can be incorporated as variation operator. This is demonstrated on a Quasi-Newton method (SR1). Based on variation, once detected local optimal solutions can be marked as niche centers and the shape of the surrounding niches is approximated from the estimated Hessian matrix. This information is used in a fitness sharing concept.

Numerical experiments indicate a reduced rate of convergence in terms of function evaluations for a parallel, asynchronous algorithm when compared
to a conventional, synchronous implementation. The use of structured populations in the form of several demes in a circular array helps to improve the solution quality over a panmictic approach at the cost of additional function evaluations. Hence, the decision to prefer the here investigated parallel, asynchronous configuration over a synchronous implementation depends on a trade-off between the speedup in runtime achievable by parallelization and the time required for these additional function evaluations.

The hybridization of the parallel, asynchronous algorithm helps to significantly improve solution quality, especially in objective landscapes with a moderate number of local optima. A significant decay can only be seen in highly multi-modal landscapes. There, the improvement of the best fitness so far as well as the additional information gained through the niche modeling by the deterministic operator requires more function evaluations than it helps to save.

Apart from the niching strategy, the method keeps the basic traits of a GA and is thus adjustable to more complex representation schemes than the here investigated linear one.

4.7 Needs for further research

The environmental selection strategy proposed here is inspired by a generational replacement and hence, does not require to have exact knowledge on individuals in the pool before the insertion of offspring. The investigation of other selection strategies, especially fitness-based approaches could help to tune selection pressure. In selection methods where parents to be replaced by offspring are explicitly picked out of the pool, concurrent worker processes have to be synchronized in some way in order to avoid race hazards. Hence, further investigations should focus on the possibilities on how such a synchronization can be realized efficiently.

The life-cycle model with a fixed $\Delta l$ allows only indirect control on the pool size. Further research could focus on more sophisticated, dynamic population models comprising effects induced by different execution times of variation operators, the current lifetime distribution in the pool, or the number of concurrent worker processes on the system.

Effects of inbreeding could be investigated. The proposed algorithm architecture with a central database allows for an exact tracking of single individuals and all operations applied to them. Hence, the information of an individual’s genealogical tree could be recorded and embedded in measures to increase population diversity or counteract premature convergence.
4.7 Needs for further research

Theoretical as well as experimental studies could investigate the speedup achievable by the proposed method. Comparisons with other parallel EA schemes like master-slave approaches or other asynchronous variants could be used to examine the advantages and limitations of the here proposed algorithm architecture.

Since the population structure is not fixed by the algorithm design, further investigations could focus on more advanced (like e.g. self-adaptive) migration schemes.

The hybridization with deterministic methods in the form of variation operators can be adapted to other local search methods from mathematical programming. If gradient information is not available, response surface methods would offer a conceptually similar approach requiring only function evaluations.

The adaption of the niche radius could be replaced by more sophisticated algorithms considering not only the objective values but as well the gradients of evaluated individuals inside a niche. Moreover, the initial guess of the niche radius could be done based on statistical measures of the iterates $x_k$ establishing the model for $B_k$.

Especially in highly non-convex functions (like Rastrigin) it could be worth to accept less accurate approximations for the local minimizers $\sigma x$ as well as the Hessian $B_k$ by using a large tolerance $\epsilon$ in the very beginning of the search. This could help to save function evaluations in non-promising regions of the search space. Later refinements could then start to reduce $\epsilon$ in order to increase the reliability of already established niche models.
Chapter 5

 Optimization of Ply Angles in Laminated Composite Structures

This chapter presents an application of the before introduced hybrid algorithm on ply angle optimization problems. An overview of ply angle optimization methods is given in Sec. 1.1.2.

5.1 Ply angle optimization: A non-linear optimization problem with periodic search space

In the following we consider scalar-valued, box-constrained ply angle optimization problems of the form:

\[
\begin{align*}
\text{minimize} \quad & F(x) \\
\text{subject to:} \quad & x \in \mathbb{R}^n \mid x_L \leq x_i < x_L + \pi
\end{align*}
\]

where \( x = [x_1, x_2, \ldots, x_n]^T \) is a vector of \( n \) ply angles. The objective \( F(x) \) is assembled from mechanical qualities of the structure like stiffness, natural frequencies, etc. These properties are evaluated by numerical simulations, i.e. finite element analysis. It is crucial for a global optimization algorithm to incorporate knowledge on the periodicity of the problem. This is achieved
by implement knowledge on the box constraints (5.2) in the optimization algorithm. Due to the geometric periodicity in the ply angles, \( F(x) \) is periodic, i.e. \( F(x) = F(x + h\pi) \) for each \( h \in \mathbb{Z}^n \). If \( F \) has at least one minimum in the constrained box (5.2) it will have infinitely many in \( \mathbb{R}^n \). The choice of periodic box-constraints on the ply angles is free to an arbitrary \( x_L \). Further on, \( x_L = -\pi/2 \) is assumed.

We assume sensitivity information to be available, i.e. \( \nabla F(x) = \frac{\partial F(x)}{\partial x} \). Sensitivities in the finite element evaluation may be used to compute \( \nabla F(x) \) at relatively low additional numerical cost (see e.g. [107, 162, 171, 180, 234]).

Ply angles are parameterized by employing the patch concept (cf. Sec. 1.1.2 and Fig. 1.5), i.e. the fiber directions in global plies. This allows to apply the method on structures constructed from locally varying laminates.

5.2 Detailing the evolution engine

The hybrid, parallel, asynchronous EA presented in Chap. 4 is employed. The following sections detail the genetic variation operators and especially their modifications for the here considered ply angle optimization problems.

5.2.1 Arithmetic Crossover

Arithmetic crossover is adapted for periodic constraints encountered in ply angle optimization problems. The basic concept is borrowed from cyclic genes and arithmetic crossovers proposed as a part of the universal genotype in [143]. Arithmetic crossover is an operation creating two offspring \( (o_1 x, o_2 x) \) from two parent individuals \( (p_1 x, p_2 x) \) by the use of a random vector \( \lambda \in \mathbb{R}^n \) \((0 < \lambda_1 < 1)\):

\[
\begin{align*}
o_1 x &= \lambda \cdot p_1 x + (1 - \lambda) \cdot p_2 x \quad (5.3) \\
o_2 x &= \lambda \cdot p_2 x + (1 - \lambda) \cdot p_1 x. \quad (5.4)
\end{align*}
\]

Thus, a conventional arithmetic crossover creates offspring in a closed, \( n \)-dimensional hypercube bounded by the alleles of the two parent solutions (offspring region 1 in Fig. 5.1). A crossover of parents with alleles \(-85^\circ\) and \(85^\circ\) for a ply angle will generate offspring with an arbitrary angle in between. Although solutions in the interval between \(85^\circ\) and \(95^\circ\) (what corresponds to \(-85^\circ\)) would inherit more information on the location of the parents and thus better fulfill the demand for a locality preserving representation. In order to generate offspring closer to the respective parents, different hypercubes in the search space are considered. Whilst in non-periodic search spaces there
exists only one such hypercube, there are infinitely many in periodic search spaces. However, these form a repetitive pattern with only $2n$ differently sized hypercubes (see Fig. 5.1 for a two-dimensional illustration). The here proposed variation operator (Alg. 11) places offspring in the smallest of these hypercubes. Hence, especially for parents close to opposite corners of the search space the here proposed method generates offspring closer to the parents (and hence inheriting more information on their geometric location) than a conventional crossover would do.

Algorithm 11 Arithmetic ply angle crossover

Require: two parent solutions $\text{p}_1x$ and $\text{p}_2x$

Ensure: two offspring solutions $\text{o}_1x = [\text{o}_1x_1, \ldots, \text{o}_1x_n]$ and $\text{o}_2x = [\text{o}_2x_1, \ldots, \text{o}_2x_n]$

1: $\tilde{x}_1 \leftarrow \text{p}_1x$
2: $\tilde{x}_2 \leftarrow \text{p}_2x$
3: $\lambda \leftarrow \text{uniform random } \in \mathbb{R}^n | 0 < \lambda_i < 1$
4: for $i \in [1, n]$ do
5: if $2\tilde{x}_i < 1\tilde{x}_i$ then
6: swap $2\tilde{x}_i, 1\tilde{x}_i$
7: end if
8: $b \leftarrow 2\tilde{x}_i - 1\tilde{x}_i$
9: $a \leftarrow \pi - b$
10: if $a < b$ then
11: $2\tilde{x}_i \leftarrow 2\tilde{x}_i - \pi$
12: end if
13: $\text{o}_1x_i \leftarrow \lambda_i \cdot 1\tilde{x}_i + (1 - \lambda_i)2\tilde{x}_i + k \cdot \pi, k \in \mathbb{Z}$ such that $-\frac{\pi}{2} \leq \text{o}_1x_i < \frac{\pi}{2}$
14: $\text{o}_2x_i \leftarrow \lambda_i \cdot 2\tilde{x}_i + (1 - \lambda_i)1\tilde{x}_i + l \cdot \pi, l \in \mathbb{Z}$ such that $-\frac{\pi}{2} \leq \text{o}_2x_i < \frac{\pi}{2}$
15: end for
16: return $\{\text{o}_1x, \text{o}_2x\}$

5.2.2 Gaussian Mutation

Gaussian mutation (Alg. 12, Fig. 5.2) is adapted to respect the cyclic box constraints of the search space. The covariance matrix $\Sigma$ is estimated from the distribution of the individuals on the parent deme. Mutated solutions violating the box-constraints are shifted by $h\pi$ (state 2 in Alg. 12).
Require: one parent solutions $p\mathbf{x}$

Require: a covariance matrix $\Sigma$

Ensure: one offspring solution $o\mathbf{x}$

1: $\bar{\mathbf{x}} \leftarrow $ random from $\mathcal{N}(p\mathbf{x}, \Sigma)$
2: $o\mathbf{x} \leftarrow \bar{\mathbf{x}} + h\pi$, $h \in \mathbb{N}$ such that $-\frac{\pi}{2} \leq o x_i < \frac{\pi}{2}$
3: return $o\mathbf{x}$

Figure 5.1: Arithmetic crossover of parents $p\mathbf{x} = [p x_1, p x_2]^T$ and $\bar{p}\mathbf{x} = [\bar{p} x_1, \bar{p} x_2]^T$ with periodic box-constraints. Offspring is generated in region 2.
Figure 5.2: Gaussian mutation of parent $\mathbf{p}_1 \mathbf{x}$ with diagonal covariance matrix $\Sigma$. The shading corresponds to a probability density.
5.2.3 SR1-mutation

As an extension to the original SR1-mutation (Sec. 4.2.7), the operator employed in this study considers the cyclic box-constraints of the search space. If an iterate $x_k$ of SR1 leaves the search space, it is shifted to $x'_k = x_k + h\pi$ with $h \in \mathbb{Z}^n$ such that $x'_k$ does no longer violate the box-constraints. Furthermore, the distance values $d(o\mathbf{x}, \mathbf{x})$ between a niche center $o\mathbf{x}$ and another individual $\mathbf{x}$ used for fitness sharing are computed as $d(o\mathbf{x}, \mathbf{x} + h\pi)$ with $h \in \mathbb{Z}^n$ such that $d$ is minimized.

5.3 Implementation

The implementation of Sec. 4.3 is extended to address structural optimization problems: The commercial finite element code ABAQUS (version 6.8) is employed for the structural simulations. Sensitivities are computed using a direct differentiation scheme (see Sec. 2.18 in [2]).

5.4 Numerical Experiments

This section presents investigations of the algorithm’s behavior in geometrically simple laminate optimization problems. In order to illustrate objectives and results the problems are restricted to two dimensions. The material data for all experiments and the examples in Sec. 5.5 are given in Tab. B.1(a) (material 1), (b) (material 2), and (c) (material 3) in Appendix B.

All experiments in this section use the same algorithm configuration: The pool size is set to 60 individuals. The population is divided into three demes. The algorithm is stopped after 4000 evaluations. The rates for the variation operators are 0.05 for Gaussian mutation, 0.05 for SR1-mutation, and 0.90 for arithmetic crossover. Migration between the demes is carried out after 600 evaluations each. Since the differentiation method applies a combined analytic/ numeric procedure, the resulting gradient information is affected by considerable numerical noise. Hence, the tolerance $\epsilon$ in SR1-mutation (i.e. the algorithm stops if $\|\nabla F(x)\| \leq \epsilon$) is set to $5 \cdot 10^{-3}$.

5.4.1 Minimal compliance design of a tensile specimen

A rectangular tensile specimen (Fig. 5.3(a), length 250mm, width 100mm, clamped at one side and uniformly loaded with 0.01N/mm at the other side) consisting of a symmetric laminate of material 1 is investigated.
The laminate is parameterized as $[\pm 30, \pm x_1, \pm x_2]_s$, whereas the $\pm 30^\circ$ plies have a thickness of 1.25mm whilst the remaining inner layers ($\pm x_1, \pm x_2$) are 0.125mm thick. The ply angles $x_1$ and $x_2$ shall be optimized in order to minimize the elastic strain energy of the loaded structure. This corresponds to the search for the stiffest configuration. A contour plot illustrating the objective as a function of the optimization variables is given in Fig. 5.3(b). From there, an obvious optimal design is given at $x = 0$, because then the fibers are oriented along the load direction. Ply angles of $90^\circ$, i.e. in $x = [0, \pm 90]$, $x = [\pm 90, \pm 90]$ or $x = [\pm 90, 0]$, obstruct the transverse contraction and hence transfer load to the outer $30^\circ$ plies what leads to stiffer designs. These can be seen as additional local optima in Fig. 5.3(b).

The runs are compared by the best fitness found after a certain number of evaluations. Hence, Fig. 5.4(a) depicts the fitness of the fittest individual in the database after $k$ evaluations with $k$ ranging from 1 to 4000 (on a logarithmic scale). Due to the stochastic components in the algorithm, the ten runs do not show the same curve. However, after 500 evaluations all runs converge to a solution of equal quality. Tab. 5.1 summarizes the fitness values,
Figure 5.4: Convergence plot for the numerical experiments: Best fitness found as a function of the number of evaluations (on a logarithmic scale) averaged over 10 runs.
5.4 Numerical Experiments

<table>
<thead>
<tr>
<th>Run</th>
<th>Best solution</th>
<th>Detected niches</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$F^*$</td>
<td>$x_1^<em>$ $x_2^</em>$</td>
</tr>
<tr>
<td>1</td>
<td>0.156032</td>
<td>0.023570 -0.140808</td>
</tr>
<tr>
<td>2</td>
<td>0.156031</td>
<td>0.008006 -0.003092</td>
</tr>
<tr>
<td>3</td>
<td>0.156031</td>
<td>-0.007092 -0.011557</td>
</tr>
<tr>
<td>4</td>
<td>0.156032</td>
<td>0.069002 0.080666</td>
</tr>
<tr>
<td>5</td>
<td>0.156031</td>
<td>-0.00620 -0.024971</td>
</tr>
<tr>
<td>6</td>
<td>0.156031</td>
<td>0.156031 -0.036184</td>
</tr>
<tr>
<td>7</td>
<td>0.156031</td>
<td>0.006691 -0.021751</td>
</tr>
<tr>
<td>8</td>
<td>0.156031</td>
<td>0.026069 -0.020761</td>
</tr>
<tr>
<td>9</td>
<td>0.156031</td>
<td>0.004075 0.004032</td>
</tr>
<tr>
<td>10</td>
<td>0.156031</td>
<td>-0.003067 -0.002079</td>
</tr>
</tbody>
</table>

avg ($F^*$) | 0.156031
std ($F^*$) | 4.21e-07

Table 5.1: Best solutions and detected niches for the tensile specimen experiment: $F^*$ in Nmm and ply angles $x_1$ and $x_2$ in degree. The niches are centered at the location given in the table head.

corresponding ply angles, and detected niches for the different runs. Only one run is capable to detect all niches (run 5). However, all runs converge in the same solution, i.e. the global optimal one. Notably, run 3 and 4 do not identify a niche there.

5.4.2 Minimal compliance design of a rectangular plate

A rectangular plate (length 500mm, width 250mm) is loaded by a concentrated, central, out-of-plane load $F = 1N$ (Fig. 5.5(a)). A symmetric laminate $[x_1, x_2]_s$ from two plies of material 1 is applied (ply thickness 0.125mm). Again, a minimal compliance design shall be found: The objective is to minimize the total strain energy. A contour plot of the objective is depicted in Fig. 5.5(b).

The best fitness averaged over all ten runs is illustrated in Fig. 5.4(b) and listed in Tab. 5.2. After 300 evaluations all runs converge to a solution of equal quality.

All runs identify both global optimal solutions. Due to the relative large tolerance in the SR1-mutation six out of ten runs identify an additional local optimum in the plateau region at $x_1 = 75$, $x_2 = 39$. 
(a) Support and loading of the rectangular plate experiment. A central out-of-plane load perpendicular to the plate surface is introduced. The plate is clamped (a) at one part of the boundary and simply supported at another (b). Positive ply angles are measured from $\xi$ to $\eta$.

(b) Contour plot of the objective for the rectangular plate experiment. The objective corresponds to the elastic strain energy of the structure (in N mm) as a function of the ply angles $x_1$ and $x_2$ (in degree). The contour lines are interpolated in a regular grid of 1296 evaluations ($5^\circ$ raster).

Figure 5.5: Geometry and objective of the rectangular plate experiment.
## 5.5 Applications

The following two examples shall illustrate the usability of the method in typical engineering applications. In contrary to the numerical examples in the section above, there is no detailed image of the objective as a function of the ply angles. Especially, there is no knowledge on the location of local or global optima. Hence, the results obtained can no longer substantiate a claim for global optimality.

### 5.4.3 Eigenfrequency optimization of a circular disk

The first natural frequency of a circular disk (radius 250mm, supported at the boundary) shall be optimized. The laminate consists of a non-symmetric laminate of two plies of material 1 (ply thickness 1mm), i.e. \([x_1, x_2]\). Since the disk geometry has no preference direction, the contour lines in Fig. 5.6 approximate straight lines (slight preference directions induced by the finite element discretization exist).

The results are illustrated in Fig. 5.4(c) and summarized in Tab. 5.3. Run 2 identified a niche at \([-8.74, -8.81]\), run 4 one at \([8.79, 8.76]\). All runs identify a solution with \(x_1 \approx x_2\).

### Table 5.2: Best solutions for the rectangular plate experiment: \(F^*\) in Nmm and ply angles \(x_1\) and \(x_2\) in degree

<table>
<thead>
<tr>
<th>Run</th>
<th>(F^*)</th>
<th>(x_1^*)</th>
<th>(x_2^*)</th>
<th>Detected niches</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.456322</td>
<td>-27.884157</td>
<td>53.289815</td>
<td>✓   ✓   ✓</td>
</tr>
<tr>
<td>2</td>
<td>0.456322</td>
<td>-27.893398</td>
<td>53.270000</td>
<td>✓   ✓   ✓</td>
</tr>
<tr>
<td>3</td>
<td>0.456322</td>
<td>27.917151</td>
<td>-53.285163</td>
<td>✓   ✓   ✓</td>
</tr>
<tr>
<td>4</td>
<td>0.456322</td>
<td>-27.905972</td>
<td>53.276329</td>
<td>✓   ✓   ✓</td>
</tr>
<tr>
<td>5</td>
<td>0.456322</td>
<td>-27.981053</td>
<td>53.285953</td>
<td>✓   ✓   ✓</td>
</tr>
<tr>
<td>6</td>
<td>0.456322</td>
<td>27.987766</td>
<td>-53.201019</td>
<td>✓   ✓   ✓</td>
</tr>
<tr>
<td>7</td>
<td>0.456322</td>
<td>27.906112</td>
<td>-53.268347</td>
<td>✓   ✓   ✓</td>
</tr>
<tr>
<td>8</td>
<td>0.456322</td>
<td>-27.897110</td>
<td>53.265182</td>
<td>✓   ✓   ✓</td>
</tr>
<tr>
<td>9</td>
<td>0.456322</td>
<td>-27.897664</td>
<td>53.269157</td>
<td>✓   ✓   ✓</td>
</tr>
<tr>
<td>10</td>
<td>0.456322</td>
<td>27.820767</td>
<td>-53.237995</td>
<td>✓   ✓   ✓</td>
</tr>
</tbody>
</table>

\[\text{avg} (F^*) \quad 0.456322\]
\[\text{std} (F^*) \quad <1e-08\]
Figure 5.6: Contour plot of the objective landscape for the disk experiment. The objective corresponds to the lowest eigenfrequency as a function of the ply angles $x_1$ and $x_2$ (in degree). The contour lines are interpolated in a regular grid of 1296 evaluations ($5^\circ$ raster) in both directions.

<table>
<thead>
<tr>
<th>Run</th>
<th>$F^\star$</th>
<th>$x_1^\star$</th>
<th>$x_2^\star$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-42.89790</td>
<td>-83.959927</td>
<td>-83.965129</td>
</tr>
<tr>
<td>2</td>
<td>-42.89800</td>
<td>-72.316472</td>
<td>-72.316472</td>
</tr>
<tr>
<td>3</td>
<td>-42.89800</td>
<td>-37.565863</td>
<td>-37.604745</td>
</tr>
<tr>
<td>4</td>
<td>-42.89780</td>
<td>-8.304790</td>
<td>-8.476144</td>
</tr>
<tr>
<td>5</td>
<td>-42.89800</td>
<td>-14.739024</td>
<td>-14.757151</td>
</tr>
<tr>
<td>6</td>
<td>-42.89800</td>
<td>-19.100151</td>
<td>-19.100151</td>
</tr>
<tr>
<td>7</td>
<td>-42.89800</td>
<td>-69.513538</td>
<td>-69.411833</td>
</tr>
<tr>
<td>8</td>
<td>-42.89800</td>
<td>-58.314708</td>
<td>-58.382630</td>
</tr>
<tr>
<td>9</td>
<td>-42.89800</td>
<td>-70.910319</td>
<td>-70.858471</td>
</tr>
<tr>
<td>10</td>
<td>-42.89800</td>
<td>-58.910033</td>
<td>-59.033110</td>
</tr>
<tr>
<td>avg ($F^\star$)</td>
<td>-42.897970</td>
<td></td>
<td></td>
</tr>
<tr>
<td>std ($F^\star$)</td>
<td>6.75e-05</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.3: Best solutions for the disk experiment: $F^\star$ corresponds to the first natural frequency $f_0$ in $Hz$, ply angles $x_1$ and $x_2$ in degree
5.5 Applications

5.5.1 Minimum compliance design of an aircraft side-rudder

An aircraft side-rudder is a control surface used to induce a yaw moment. It is attached to the aircraft’s vertical tail and allows for rotations around the aircraft’s yaw axis (Fig. 5.7). The steering moments are induced into the bottom rib, a second attachment point is formed by two short supports on the eighth rib. The rudder is subject to aerodynamic forces acting on its skin. An additional, non-structural mass is positioned in the front top section of the structure to balance steering reactions. Depending on the flight state, this mass induces considerable forces of inertia. The structure is built from a single spar and several ribs supporting the aerodynamic active skin. The rounded nose allows for a gap-free connection to the vertical tail. The trailing edge gives support to a trim-rudder. The component is nearly symmetric with respect to the \( x-z \)-plane. The slight asymmetry is induced by the trailing edge which is inclined to the port side to give place for the trim-rudder attachment and actuation devices.

For the analysis the different distributed and concentrated loads of a dimensioning flight state are reduced to a set of line-distributed forces and moments (Shear-Moment-and-Torque (SMT)) on the wing-box centerline. From there, the loads are summed up to the rib positions and introduced into each rib’s center point which is connected to the rib’s boundary by rigid body elements (cf. e.g. Appendix A in [155] for an overview of the SMT-method). The resulting displacement field for the considered load case is a superposition of a torsional deformation around the \( z \)-axis and two-point bending between the support points.

The structure consists of a ground laminate and shall be symmetrically reinforced by five non-overlapping patches on the skin (Fig. 5.8). The internal structure (ribs, spar, and trailing edge) consist of a quasi-isotropic laminate (Material 1, thickness 1.5mm). The remaining laminates are all built from plies of material 1 of 0.125mm thickness: The nose (section 5 in Fig. 5.8(b)) is covered by a laminate \([\pm x_1, x_2]_s\). The skin is covered by a laminate \([\pm x_3]_s\). This laminate is reinforced by additional plies, i.e. \([\pm x_3, \pm x_4]_s\) in section 1, \([\pm x_3, \pm x_5]_s\) in section 2, \([\pm x_3, \pm x_6]_s\) in section 3, and \([\pm x_3, \pm x_7]_s\) in section 4 respectively.

The optimization is carried out to maximize the stiffness, i.e. to minimize the elastic strain energy. The decision variables are the seven ply angles \([x_1, x_2, \ldots, x_7]^T\).
Figure 5.7: Aircraft side rudder structure. The rudder is attached to the vertical tail whereas the upper attachment point enables for rotations around the vertical axis. The lower support is used to introduce the steering moment and is hence fixed in the analysis.
5.5 Applications

Figure 5.8: Laminate on the aircraft side rudder: An existing ground laminate on the skin is reinforced with five non-overlapping patches. The ply angles are measured from $\xi$ to $\eta$.

Results and discussion: The optimization is carried out for 10000 function evaluations. All evaluations are plotted in Fig. 5.9. SR1-mutation identifies 29 niches. The best individual found has an objective value $F(x^*) = 436.558\,N\,mm$ with $x^* = [-25.859, 25.679, -57.405, 0.303, -57.461, 59.837, 49.077]^T$. It is created after 5131 evaluations as offspring of SR1-mutation. Its gradient $\nabla F(x^*)$ evaluates to $[-3.450, 3.803, 14.0219, 6.386, -1.798, -4.666, 1.301]^T \cdot 10^{-4}$.

The ply angles $x_1$ and $x_2$ reinforce the wing nose box against torsion deflections. The same holds for the skin $x_3$ and patches 3 ($x_6$) and 4 ($x_7$). Patch 1 ($x_4$) is affected by the fixed support in the lowest rib. Ply angle optimization is able to increase the stiffness of the structure by roughly 5% when compared to the best solution in the initial population.

5.5.2 Eigenfrequency optimization of a racing car rear wing

A racing car rear wing (Fig. 5.10) is a load carrying structure designed to exert an aerodynamic down force on the rear wheels of the car to maximize traction. At the same time, a car’s weight shall be minimized. Harmonic vibrations of the rear wing reduce the aerodynamic performance and may, if in transverse direction, even cause collisions with carriage components. Hence, the most critical lowest eigenfrequency shall be maximized. The transverse
Figure 5.9: Convergence plot for the side rudder optimization subdivided to variation operators. Each plot point (marked by a cross) corresponds to a single evaluation. The solid line indicates the best solution found so far.
Figure 5.10: Racing car rear wing consisting of the components side plate \((A)\), lower wing \((B)\), upper wing \((C)\), top wing \((D)\), and ribs \((E)\). All degrees of freedom are set to zero at the interface to the car body (for all nodes on the edge adjacent to the shaded region).

eigenmodes are mainly dominated by the stiffness of the side plates and their connection to the car body, i.e. the lower wing. Hence, ply angles of reinforcement plies in the side plates and the lower wing shall be optimized. The side plates consist of four plies of material 3, ply thickness 0.25 mm) with angles \([0, 45]_s\) locally reinforced on both sides by four patches of material 2, ply thickness 0.2 mm, geometry illustrated in Fig. 5.11(a)-5.11(d)) with orientations \([x_1, x_2, x_3, x_4]\). The lower wing consists of a symmetric laminate from material 3 \([90, \pm 45, 0]_s\) (ply thickness 2 mm each), reinforced on both sides by four additional plies (thickness 0.4 mm each) of material 2 with ply angles \([\pm x_5, \pm x_6]\). The remaining components (upper wing, top wing, and ribs) are built from a laminate of material 3 \([0, \pm 45, 90]_s\) with a ply thickness of 0.25 mm each.

The optimization is carried out to maximize the lowest eigenfrequency which is equivalent to minimizing its negative, i.e.

\[
\text{minimize } F(x) = -f_1 
\]

with six ply angles \(x = [x_1, \ldots, x_6]^T\) as decision variables.

**Results and discussion:** The optimization is carried out for 10000 function evaluations. All evaluations are plotted in Fig. 5.12. SR1-mutation identifies 33 niches. The best individual found has an objective value
Figure 5.11: Racing car rear wing laminate patches: The component is symmetric with respect to the $x$-$z$-plane. The ply angles are oriented for the left car side ($y > 0$) according to the coordinate systems indicated by $\xi$ and $\eta$ (positive ply angles are measured from $\xi$ to $\eta$). The laminate properties for the right side are mirrored at the $x$-$z$-plane.

\[
F(x^*) = -15.719 \text{ with } x^* = [-39.206, 7.746, -64.208, -80.218, 0.924, 65.419]^T.
\]

It is created after 2722 evaluations as offspring of an arithmetic crossover. Its gradient $\nabla F(x^*)$ evaluates to $[-0.386, 31.369, -1.143, -8.908, -12.617, -4.381]^T \cdot 10^{-4}$.

Fig. 5.12 illustrates properties of SR1-mutation as sequences of locally converging evaluations (e.g. a prominent sequence converging in the range of $F = -12.4$). These local searches are carried out concurrently and find different sub-optimal solutions. However, Fig. 5.12 is not able to illustrate the contemporaneity of the different evaluations. The improvement of the optimal design is about 1.5% when compared with the best solution in the initial pool.

The ply angles $x_1$, $x_3$, and $x_4$ are negative and thus reinforce the side plates in direction from the lower wing towards the upper wing. This can be interpreted as a reinforcement against a transverse bending mode of the structure. Patch 2 (ply angle $x_2$) reinforces in-plane shear deformations. In the lower wing $x_5$ reinforces the critical bending direction of the transverse bending mode. The ply angle $x_6$ reinforces the lower wing against bending and shear deformations.

### 5.6 Concluding remarks

A method aimed at the global optimization of ply angles in laminated composites has been proposed. The method builds on a hybrid (first-order), niching EA. Genetic variation operators are tailored to the special properties of ply
Figure 5.12: Convergence plot for the rear wing optimization apportioned to variation operators. Each plot point (marked by a cross) corresponds to a single evaluation. The solid line indicates the best solution found so far.
angle optimization problems. Cyclic box-constraints are considered in the crossover, mutation, and niching operations.

A direct parameterization of ply angles of global plies is attractive due to its simplicity. It allows for an easy interpretation of the results of an optimization and for the incorporation of manufacturing considerations in the form of constraints on these ply angles. However, the direct parameterization of ply angles imposes the risk to cope with multi-modal objectives. Nonetheless, these objectives are usually smooth and differentiable at acceptable numerical cost. The application of only deterministic searches would probably get stuck in local optimal solutions. EAs have a better chance to obtain global optimal solutions at considerably increased numerical cost. Thus, the here investigated hybrid algorithm is aimed at combining the strength of both approaches, i.e. to use gradient information and information on already discovered local optimal solutions to reduce the number of function evaluations in a stochastic EA.

There is no guarantee for the method to find a global optimal solution. Nonetheless, in all 30 examined optimization runs in three simple academic test cases the method identified a global optimal solution. It keeps book of all visited solutions in a database. This is particularly relevant, if several optimal designs of equal quality exist. Then, knowledge on these can help to establish a better understanding of a specific structural optimization problem which in turn may lead to improved structural concepts.

The method is applicable to a wide range of problems, i.e. non-convex objectives, geometries, and loadings. Constrained problems may be solved by penalty methods (cf. e.g. Sec. 2.4.2). Two application examples illustrate the capabilities of the approach on more complex structures. There, the method showed its ability to escape from local optimal solutions.

The method has been investigated in small and medium sized problems. Larger search spaces have not been examined. Larger problems will obviously expose more complex objective landscapes. Additionally, the time required for the processing of derivative information and the communication in between the different subalgorithm will rise as well. Notably, the timings depend considerably on the actual implementation and the system configuration. Therefore, one has to accept a tradeoff between the runtime gained by hybridization and the time lost in the overhead it induces. This tradeoff will probably give foundation for an upper limit on an affordable problem size. Further research could theoretically and experimentaly determine such a limit.

Prior work (e.g. [89, 262]) illustrates the possibility to optimize discrete ply angles in EA-based optimizations. The choice for discrete ply angles is mainly motivated by manufacturing restrictions one one hand and by nar-
rowing the search space for a faster convergence on the other hand. Notably, many industrial applications consider only a small subset of ply angles (like for instance $0^\circ$, $\pm 45^\circ$, and $90^\circ$). Obviously, such restrictions may considerably reduce the achievable performance of a structure when compared to continuous ply angles. The choice for discrete ply angles prohibits the direct use of gradient information in the here investigated way. Nonetheless, the investigation of methods to make use of derivative information in discrete ply angle optimization problems could be a topic for further research.
98 Optimization of Ply Angles in Laminated Composite Structures
Chapter 6

Optimization of Laminated Composites on Geometrically Partitioned Shells

This chapter investigates a genetic representation aimed at the automated design of laminated composites. It builds on the encoding of global plies. Special attention is payed to the representation of the plies’ shape attribute. An overview on laminate optimization methods is given in Sec. 1.1.2. The here presented method borrows graph concepts from [93], [89], and [133]. In order to reduce the runtime of a single optimization run, sensitivity information and local searches are embedded in the framework of a parallel, asynchronous EA scheme.

6.1 Genetic representation and variation of laminated composites

6.1.1 A laminate optimization problem formulated on sections

Investigations outlined in Sec. 1.1.2 concentrate on methods to map global plies to laminates. The optimization problem in this study is formulated
in the opposite direction: given a shell-like structure partitioned to a set of geometrical regions (further on called sections) $S$, find the number and sequence and associated attributes of patches minimizing a criterion $F$. Each patch covers a connected subset of $S$ defining its shape. The connectivity of sections in $S$ can be determined from their geometrical adjacency, i.e. sections sharing geometric edges are adjacent.

As already proposed in the references cited above, graph operations enable for an elegant way to formulate the optimization problem. Therefore, we introduce a section-graph $\mathcal{G} = (S, E)$ as an abstract model of the structure to optimize. The section-graph $\mathcal{G}$ is an undirected graph with the set of sections $S$ building its vertices. Geometrically adjacent sections are connected by edges $E$ (see Fig. 6.1).
Further on, we consider the \( i^{th} \) patch to be defined by its shape \( P_i \), a material \( m_i \) from a list of available materials, an orientation in the form of a rotation angle \( \phi_i \) with respect to a reference orientation, and a thickness factor \( t_i \). The actual thickness of the patch then corresponds to \( t_i \) multiplied by the raw thickness of the ply material \( m_i \). This is a concession to manufacturing, where the thickness of a reinforcement treatment can only vary in discrete steps.

Hence, the laminate optimization problem can be formulated as:

\[
\min_{x \in \{P_i, m_i, \phi_i, t_i\}^n} F(x) \tag{6.1}
\]

subject to:

\[
\begin{align*}
    n &\in \mathbb{Z}, n_{\text{min}} \leq n \leq n_{\text{max}} & \text{(number of patches)} \\
    P_i &\subseteq G, P_i \text{ connected} & \text{(subgraph of } G) \\
    m_i &\in \{M_0, \ldots, M_m\} & \text{(ply material)} \\
    \phi_i &\in \mathbb{R}, -90^\circ < \phi_i \leq 90^\circ & \text{(ply angle)} \\
    t_i &\in \mathbb{Z}, 1 \leq t \leq t_{\text{max}} & \text{(ply thickness factor)}
\end{align*}
\]

the representation is of *variable-length*, i.e. the number of dimensions \( n \) in the search space is not a priori defined but a variable to be optimized. The search space is *structured* in blocks of equal form, namely tuples \( \langle P_i, m_i, \phi_i, t_i \rangle \) representing one reinforcement patch each. The sequence of patches in \( x \) is *sensitive to ordering*, since its order determines the stacking sequence in the laminate. The patch attributes are *heterogeneous*, i.e. graph-, string-, real-, and integer-valued.

An analogous, fixed-length problem could be found by introducing \( n_{\text{max}} \) virtual patches and parameterizing their physical existence either by an additional boolean value or by allowing a zero thickness. As shown in [89], this approach would suffer from a potentially large number of variables representing attributes of physically inexistent patches. These so-called non-coding regions participate in the optimization process without affecting the objective. Hence, they affect the performance of the optimization method especially if there is a considerable difference between \( n_{\text{min}} \) and \( n_{\text{max}} \).

For the following discussion we introduce the notation \( |\cdot| \) for the length of an arbitrary list and \( |x|_P \) for the number of plies in a candidate solution. The \( i^{th} \) ply tuple is called \( x_i \).
Figure 6.2: Illustration of different types of sections in a structure with non-regular sections and a single patch.
6.1.2 Section categories

In order to develop operations modifying the shape of single patches, we characterize the sections in the patch graph $P_i$ of a single patch $i$ (see Fig. 6.2). Therefore, the notation for some common graph functions is introduced: $v(G)$ returns the set of vertices in a graph $G$, $\text{v}_{\text{adj}}(v, G)$ returns the set of vertices adjacent to vertex $v$ in a graph $G$, and $v_{\text{art}}(G)$ returns the articulation vertices (also called cut vertices) of a graph $G$. The set of patch sections, i.e. all sections contained in the $i^{th}$ patch is written as:

$$\{v_{\text{patch}}\} = \{v \in v(G) : v \in v(P_i)\}$$

(6.7)

The set of sections adjacent to patch $i$ can be written as

$$\{v_{\text{neighbor}}\} = \{v \in v(G) : v \notin v(P_i), \text{v}_{\text{adj}}(v, G) \cap v(P_i) \neq \varnothing\}$$

(6.8)

And the set of section on the boundary of it is computed from

$$\{v_{\text{boundary}}\} = \{v \in v(G) : v \in v(P_i), \text{v}_{\text{adj}}(v, G) \cap \{v_{\text{neighbor}}\} \neq \varnothing\}$$

(6.9)

The set of articulation sections $\{v_{\text{articulation}}\}$ is a subset of the boundary sections: The removal of one articulation section splits $P_i$ into two or more components.

$$\{v_{\text{articulation}}\} = v_{\text{art}}(P_i)$$

(6.10)

6.1.3 Section rankings with gradient information

Gradient information is used to indicate whether a section’s thickness has to be increased or decreased. Therefore, each section $j$ in $S$ is accompanied by a scalar parameter $\tau_j$ called the section’s thickness multiplier. In the analysis model the ply thickness of each ply covering section $j$ is multiplied by $\tau_j$ which finally multiplies the laminates thickness by $\tau_j$. By setting the section multiplier $\tau_j$ to 1 it does not actually change the physical expression of the laminate. Nonetheless, it allows for the computation of the derivative $\nabla \tau F(x) = [\partial F(x) / \partial \tau_j]$. The number of sections is constant during the optimization and typically considerably smaller than the number of finite elements in the analysis model. The computation of these derivatives can be done by sensitivity analysis at an affordable additional cost ([171, 170, 162, 248]).
The information available from these section thickness derivatives is injected into genetic operators locally modifying the laminate. For the further discussion we introduce rank \((e, e)\) as an expression for the index of item \(e\) in a sequence ordered in ascending order. \(\text{rank}(e, e)\) is an integer number in the interval \([0, |e| - 1]\). A rank value of zero indicates that \(e\) has the lowest value in \(e\) whilst a rank \(|e| - 1\) designates the highest value. The section rank of section \(j\) is then computed as:

\[
\rho_j = \text{rank}\left(\frac{\partial F(x)}{\partial \tau_j}, \nabla \tau F(x)\right)
\]  

Sections with low (thus potentially negative) thickness derivatives have low ranks \(\rho_j\). Hence, increasing the thickness of a section with low rank will lower (or at least only little increase) the objective of a candidate solution \(x\). The same holds for decreasing the thickness of sections with high rank.

Thus, section rankings can indicate a tendency on where reinforcements in the form of the here considered patches shall be applied or where these reinforcements shall be removed. However, they depend on the local laminate already existing in the corresponding section. Thus, section rankings may miss-estimate the influence of a reinforcement treatment if there are significant differences in the homogenized properties of the existing laminate and a potential reinforcement patch. Hence, section rankings are embedded in stochastic operations. There, they shall bias operations modifying the patch shapes without affecting the global search properties of the EA scheme.

### 6.1.4 Basic operations on the patch shape

Based on the above introduced section rankings basic operations modifying patch shapes can be developed. *Add one section* (Alg. 13) adds a new neighbor section to an existing patch shape. The choice on which section to add is based on section rankings. The exponent in the power law (state 3) influences the balance between a random and a deterministic choice: an exponent of zero corresponds to a uniform random selection. For this study the exponent is set to 5.

*Remove one section* (Alg. 14) removes a boundary section from an existing patch shape. Articulation sections are not considered for removal in order to keep the patch graph connected. The choice on which section to remove is again based on section rankings. The power law (state 5) to compute the selection weights is complementary to the one in Alg. 13.
Algorithm 13 Add one section

**Require:** section rankings \( \varrho = \{ \varrho_j \} \)

**Require:** a patch shape (patch-graph) \( \mathcal{P} \)

**Ensure:** a modified shape \( \mathcal{P}' \)

1: \( \{ v_{\text{neighbor}} \} \leftarrow \text{neighbor sections of} \ \mathcal{P} \) (see (6.8))
2: \( m \leftarrow |v_{\text{neighbor}}| \)
3: \( w \leftarrow \left\{ \left( \frac{m-\varrho_j - 1}{m - 1} \right)^5 \ \forall \text{sections} \ j \in v_{\text{neighbor}} \right\} \)
4: \( s \leftarrow \text{weighted random choice from} \ v_{\text{neighbor}} \ \text{with weights} \ w \)
5: \( \mathcal{P}' \leftarrow \text{add section} \ s \ \text{as a new vertex to} \ \mathcal{P} \)

Algorithm 14 Remove one section

**Require:** section rankings \( \varrho = \{ \varrho_j \} \)

**Require:** a patch shape (patch-graph) \( \mathcal{P} \)

**Ensure:** a modified shape \( \mathcal{P}' \)

1: \( \{ v_{\text{boundary}} \} \leftarrow \text{boundary sections of} \ \mathcal{P} \) (see (6.9))
2: \( \{ v_{\text{articulation}} \} \leftarrow \text{articulation sections of} \ \mathcal{P} \) (see (6.10))
3: \( \{ v_{\text{removable}} \} \leftarrow v_{\text{boundary}} \setminus v_{\text{articulation}} \)
4: \( m \leftarrow |v_{\text{removable}}| \)
5: \( w \leftarrow \left\{ \left( \frac{\varrho_j}{m - 1} \right)^5 \ \forall \text{sections} \ j \in v_{\text{removable}} \right\} \)
6: \( s \leftarrow \text{weighted random choice from} \ v_{\text{removable}} \ \text{with weights} \ w \)
7: \( \mathcal{P}' \leftarrow \text{remove section} \ s \ \text{from} \ \mathcal{P} \)
6.1.5 Genetic variation operators

The aforementioned traits of the here investigated genetic representation influence the choice and design of variation operators applicable to it. Since it is structured on two different hierarchical levels – namely a laminate level and a ply level – there have to be variation operators changing information on both levels. A set of only two variation operators like in conventional EAs would hardly enable for non-disruptive changes in all attributes of parent solutions (cf. local and Lamarckian property in Sec. 2.3). Thus, there is a set of different genetic operations for each level and attribute-type.

The following operators modify information on the laminate level: Stacking mutation changes the stacking sequence in a laminate Alg. 15. Add ply mutation adds a new ply to the laminate (Alg. 16). Remove ply mutation removes a ply from the laminate (Alg. 17). Split-and-splice ply crossover cuts the laminate stack of two parent solutions each at a random position and recombines the head and tail segments to two new offspring (cf. Chap. 3).

**Algorithm 15 Stacking mutation**

| Require: | one parent solution $p_x = \{p_{x0}, \ldots, p_{xn-1}\}$ |
| Ensure:  | one offspring solution $o_x$ |
| 1: $p \leftarrow$ random integer in $(1, n - 1)$ |
| 2: $o_x \leftarrow \{p_{xp}, p_{xp+1}, \ldots, p_{xn-1}, p_{x0}, p_{x1}, \ldots, p_{xp-1}\}$ |
| 3: return $o_x$ |

The following operators act on the ply level: Material mutation selects a random ply from a parent solution and creates one offspring by randomly exchanging the material attribute of the selected ply. Gaussian ply-angle mutation changes the ply angles of all plies in a parent solution by adding a random value from $N(\mu = \phi_i, \sigma = 5^\circ)$. SR1-ply-angle mutation uses a symmetric rank-1-updating to locally optimize the ply angles in a parent solution (see Chap. 5). Ply-thickness mutation sets the ply thickness of a random ply in a parent solution to a random integer in $[1, t_{max}]$. Grow-ply mutation selects a random ply in a parent solution and adds an additional section to the patch shape by Alg. 13. Shrink-ply mutation selects a random ply in a parent solution and removes a section from the patch shape by Alg. 14. Move-ply mutation selects a random ply in a parent solution. Then the patch shape is modified by first applying Alg. 13 and later Alg. 14. Thus, it adds one section and removes potentially another one, i.e. it changes the location of reinforcement patch. Ply-property crossover recombines ply attributes in two parent solutions (Alg. 18). This operator exchanges ply materials and employs arithmetic crossovers on ply-thicknesses (see e.g. [143]) and ply-angles.
Algorithm 16 Add ply mutation

Require: one parent solution \( p^x = \{ p_{x_0}, \ldots, p_{x_{n-1}} \} \)

Require: section rankings \( \varrho = \{ \varrho_j \} \) in \( p^x \)

Ensure: one offspring solution \( o^x \)

1: \( p \leftarrow \text{random integer in } (1, n-1) \)
2: \( m \leftarrow \text{random element from } \{ M_0, \ldots, M_m \} \)
3: \( \phi \leftarrow \text{uniform random real in } (-90, 90] \)
4: \( t \leftarrow \text{uniform random integer in } [1, t_{\text{max}}] \)
5: \( v \leftarrow v(G) \)
6: \( l \leftarrow |v| \)
7: \( w \leftarrow \left\{ \left( \frac{l-\varrho_j-1}{l-1} \right)^5 \right\}_{\forall \text{ sections } j \in v} \)
8: \( s \leftarrow \text{weighted random choice from } v \text{ with weights } w \)
9: \( P \leftarrow \text{initialize patch graph with one section } s \)
10: \( k_{\text{max}} \leftarrow \text{number of sections on connected component of } G \text{ containing } s \)
11: \( k \leftarrow \text{random integer in } [1, k_{\text{max}}] \)
12: while \( |v(P)| < k \) do
13: \( \text{add one section to } P \) (Alg. 13)
14: end while
15: \( x \leftarrow \langle P, m, \phi, t \rangle \) (random ply)
16: \( o^x \leftarrow \{ p_{x_0}, p_{x_1}, \ldots, p_{x_{p-1}}, x, p_{x_p+1}, \ldots, p_{x_{n-1}} \} \)
17: return \( o^x \)

Algorithm 17 Remove ply mutation

Require: one parent solution \( p^x = \{ p_{x_0}, \ldots, p_{x_{n-1}} \} \)

Require: section rankings \( \varrho = \{ \varrho_j \} \) in \( p^x \)

Ensure: one offspring solution \( o^x \)

1: for \( i \in [0, |p^x|] \) do
2: \( \omega_i \leftarrow \sum_{j \in v(P_i)} \left( \frac{\varrho_j}{m-1} \right)^5 \)
3: end for
4: \( p \leftarrow \text{weighted random integer in } [0, |p^x|] \) according to weights \( w = [w_i] \)
5: \( o^x \leftarrow \{ p_{x_0}, p_{x_1}, \ldots, p_{x_{p-1}}, p_{x_p+1}, \ldots, p_{x_{n-1}} \} \)
6: return \( o^x \)
Optimization of Lam. Composites on Geom. Part. Shells

Fig. 6.3 illustrates the plies participating in the crossover if the two parents have not the same length.

**Algorithm 18** Ply-property crossover

**Require:** two parent solutions $\mathbf{p}_1 \times = \{p_{1,x_0}, \ldots, p_{1,x_{n_1-1}}\}$ and $\mathbf{p}_2 \times = \{p_{2,x_0}, \ldots, p_{2,x_{n_2-1}}\}$

**Ensure:** two offspring solutions $\mathbf{o}_1 \times, \mathbf{o}_2 \times$

1. $\mathbf{o}_1 \times \leftarrow \mathbf{p}_1 \times$
2. $\mathbf{o}_2 \times \leftarrow \mathbf{p}_2 \times$
3. $s \leftarrow$ uniform random section from $v (G)$
4. $\mathbf{o}_1 \times' \leftarrow$ all plies in $\mathbf{o}_1 \times$ covering $s$
5. $\mathbf{o}_2 \times' \leftarrow$ all plies in $\mathbf{o}_2 \times$ covering $s$
6. $m \leftarrow \min (|\mathbf{o}_1 \times'|, |\mathbf{o}_2 \times'|)$
7. $1 \mathcal{I} \leftarrow \{i \in [0, |\mathbf{o}_1 \times'|) : i \mod \lfloor |\mathbf{o}_1 \times'|/m \rfloor = 0 \}$ \{index set of crossover plies in first parent\}
8. $2 \mathcal{I} \leftarrow \{i \in [0, |\mathbf{o}_2 \times'|) : i \mod \lfloor |\mathbf{o}_2 \times'|/m \rfloor = 0 \}$ \{index set of crossover plies in second parent\}
9. $\mathbf{o}_1 \times'' \leftarrow \{o_{1,i}' \forall i \in 1 \mathcal{I}\}$
10. $\mathbf{o}_2 \times'' \leftarrow \{o_{2,i}' \forall i \in 2 \mathcal{I}\}$
11. for $k \in [0, m)$ do
12. \hspace{1em} $a \leftarrow 1 \mathcal{I}_k$ \{index of first crossover ply\}
13. \hspace{1em} $b \leftarrow 2 \mathcal{I}_k$ \{index of second crossover ply\}
14. \hspace{1em} swap $(\lambda o_{a''} m_{a''}, \lambda o_{b''} m_{b''})$ \{exchanges materials between plies\}
15. \hspace{1em} crossover $(\lambda o_{a''} t_{a''}, 2 \mathcal{I}_k)$ \{arithmetic crossover of ply thicknesses\}
16. \hspace{1em} crossover $(\lambda o_{a''} \phi_{a''}, 2 \mathcal{I}_k)$ \{arithmetic crossover of ply angles\}
17. end for
18. return $\{\mathbf{o}_1 \times, \mathbf{o}_2 \times\}$

**6.2 Implementation**

The implementation borrows from the one presented in Sec. 5.3 and 4.3. The commercial finite element code ABAQUS (version 6.8) is employed for the structural simulations. Sensitivities are computed by a direct differentiation scheme (see Sec. 2.18 in [2]). The implementation operates on ABAQUS-input files and recognizes user defined element sets as sections. The section-graph is constructed automatically from meshed shell structures with arbitrary geometry. A set of naming conventions and regular expressions is employed to
map candidate solutions back to valid ABAQUS-input files. Special mapping procedures for geometrically symmetric structures, regions excluded from optimization, or symmetric laminates are available. The lamination plan of each candidate solution is fully parameterized in the corresponding input file. The dependencies and – if available – symbolical derivatives of dependent parameters are constructed automatically and passed to the ABAQUS preprocessor.

6.3 Numerical Experiments

6.3.1 Single-ply vibrating plate

A rectangular plate (Fig. 6.4) with an additional, non-central mass shall be optimized for its first natural frequency $f_0$, subject to an upper limit constraint on the overall structural mass $\mathcal{M} \leq \mathcal{M}_{\text{max}}$. The structure is covered by a very compliant fill-material (isotropic, Young’s modulus $E = 1\,\text{GPa}$, Poisson’s ratio $\nu = 0.3$, thickness 0.15$\text{mm}$, area density 0.237$\text{kg/m}^2$). The shaded region in Fig. 6.4 is covered by an additional mass layer (isotropic, $E = 1\,\text{GPa}$, $\nu = 0.3$, thickness 10$\text{mm}$, area density 158$\text{kg/m}^2$). In a first experiment, the structure shall be reinforced by a single, additional layer of an unidirectional carbon/epoxy prepreg (Tab. B.1(a), thickness 0.15$\text{mm}$, area density 0.237$\text{kg/m}^2$). The experiment corresponds to problem 5.3.1 presented in [89] (revisited in Sec. 7.5 in the same source and also presented in [93]).

The original optimization problem is formulated as:

$$\max_x f_0 (x)$$

subject to $\mathcal{M} (x) - \mathcal{M}_{\text{max}} \leq 0$ \hspace{1cm} (6.12)

An exterior penalty method and scaling is employed in order to get an unconstrained, scalar valued fitness function (see Sec. 2.4.2):

$$\min_x F (x) = -\frac{f_0 (x)}{f_s} + p \left[ \max \left\{ 0, \frac{\mathcal{M} (x) - \mathcal{M}_{\text{max}}}{\mathcal{M}_s} \right\} \right]^2$$ \hspace{1cm} (6.14)
Figure 6.4: Geometry of the vibrating plate experiment: The rectangular plate is simply supported on all edges. The shaded region corresponds to an additional mass. All dimensions are in mm.

Figure 6.5: Results of the single-ply vibrating plate experiment

On the constraint (i.e. when $M(\mathbf{x}) = M_{\text{max}}$) the limit value from the forbidden side of the constraint is taken as derivative of $F$ with respect to decision variables in $\mathbf{x}$. For the here investigated problem, the values are set to $M_{\text{max}} = 0.436475\, \text{kg}$, $M_s = 0.05\, \text{kg}$, $p = 10$, and $f_s = 2\, \text{Hz}$. The number of plies ($n$) is restricted to one ($n_{\text{min}} = n_{\text{max}} = 1$), the ply material list contains only the above characterized unidirectional prepreg, and the number of raw material layers in the reinforcement is set to one ($t_{\text{min}} = t_{\text{max}} = 1$). An unbiased setup of fifty sections in a regular grid of five along the short and ten along the long side of the plate (cf. Fig. 6.6) is chosen. The finite element model contains 200 four-node, layered-shell elements in a square shape, thus, each section is modeled by four elements.

Ten independent optimization runs are carried out. For each run, the
6.3 Numerical Experiments

(a) $f_0 = 2.042\, Hz$, $M = 0.436\, kg$, $\phi = -88.51^\circ$

(b) $f_0 = 2.042\, Hz$, $M = 0.436\, kg$, $\phi = 88.36^\circ$

(c) $f_0 = 2.036\, Hz$, $M = 0.436\, kg$, $\phi = 89.84^\circ$

(d) $f_0 = 2.042\, Hz$, $M = 0.436\, kg$, $\phi = 87.91^\circ$

(e) $f_0 = 2.046\, Hz$, $M = 0.436\, kg$, $\phi = 88.23^\circ$

(f) $f_0 = 2.040\, Hz$, $M = 0.436\, kg$, $\phi = -89.92^\circ$

(g) $f_0 = 2.042\, Hz$, $M = 0.436\, kg$, $\phi = -87.90^\circ$

(h) $f_0 = 1.928\, Hz$, $M = 0.436\, kg$, $\phi = 89.95^\circ$

(i) $f_0 = 2.047\, Hz$, $M = 0.436\, kg$, $\phi = -88.62^\circ$

(j) $f_0 = 2.042\, Hz$, $M = 0.436\, kg$, $\phi = 88.15^\circ$

Figure 6.6: Best individuals identified in 10 different runs for the single-ply vibrating plate experiment with their first eigenfrequency $f_0$, mass $M$, and ply-angle $\phi$. 
pool size is set to 250 individuals, 4 demes are employed, and migration frequency is set to 750 evaluations. The application rates for the operators are set to 90% for ply-property crossover, 2% for Grow-ply-mutation, 2% for Shrink-ply-mutation, 2% for Move-ply-mutation, 2% for Gaussian ply-angle mutation, 2% SR1-ply-angle mutation, and zero for the remaining operators. The optimization is stopped after 15000 evaluations.

**Results and discussion:** Convergence plots for the first eigenfrequency and the mass are given in Fig. 6.5. There, it can be seen that from around 7000 evaluations on, no run is able to improve its best solution. As can be expected, all allowed mass is invested in a reinforcement layer and hence all solutions found are on the constraint. Moreover, nine out of ten runs find a solution with an eigenfrequency in a range between 2.036 and 2.047 Hz, only one run gets stuck at 1.928 Hz.

The shape of the single reinforcement patch for the best individual of each run is shown in Fig. 6.6. Although the geometry of the structure has a horizontal symmetry, only one reinforcement layer shows this symmetry (Fig. 6.6(h)) with a symmetric patch shape and a ply-angle of approximately 90°. Interestingly, this solution has the lowest eigenfrequency within the ten best solutions. Given the horizontal symmetry, six out of ten solutions have the same patch shape (Fig. 6.6(a), 6.6(b), 6.6(d), 6.6(f), 6.6(g), and 6.6(j)). Their ply-angles are in range of 87.91° to 89.92° and the corresponding eigenfrequency ranges from 2.040 to 2.042 Hz. The two best solutions (Fig. 6.6(e) and 6.6(i)) feature a hole in the patch shape where the mass layer is situated. This allows for making use of the stiffness of the mass layer and at the same keeping the mass in this critical region at the lowest value possible. Since there is no operator inserting holes in a patch shape, the patch has just grown around the mass section for these two runs.

### 6.3.2 Twelve-ply vibrating plate

The vibrating plate experiment is repeated with the following changes: A variable number of patches is allowed in an interval $1 \leq n \leq 12$. The maximum ply thickness factor is set to $t_{\text{max}} = 3$. The fitness function (Eq. 6.14) is adapted to the heavier structure by setting $f_s = 30 Hz$, $M_{\text{max}} = 0.543125kg$, and $M_s = 0.05 kg$.

The variation operator rates are set to: 50% one-point ply crossover, 40% ply-property crossover, and 1% for each of the remaining operators. In order to cope with the larger search space the pool size is increased to 400 individuals.
Results and discussion: Convergence plots are given in Fig. 6.7. The plies of the best solution of all runs (run 2) are illustrated in Fig. 6.8. The best solution reaches an eigenfrequency of $28.214 \text{Hz}$ at a mass of $0.543125 \text{kg}$. The remaining runs find a solution with $f_0 = 24.7396$, $M = 0.543125 \text{kg}$ and $f_0 = 24.9772 \text{Hz}$, $M = 0.53957 \text{kg}$ respectively.

The convergence plots in Fig. 6.7 show little improvements over the last evaluations but not yet a stagnation after 100000 function evaluations. Hence, the runs may not be fully converged. As expected, the mass constraint is reached by all solutions. The patch shapes as well as the angles shown in
Figure 6.8: Plies of the best individual for the twelve-ply vibrating plate experiment with their corresponding ply angles $\phi$ and thickness multipliers $t$. 

(a) Ply 1 : $\phi = 64.99^\circ$, $t = 2$

(b) Ply 2 : $\phi = -87.31^\circ$, $t = 1$

(c) Ply 3 : $\phi = -71.47^\circ$, $t = 1$

(d) Ply 4 : $\phi = -72.93^\circ$, $t = 2$

(e) Ply 5 : $\phi = -75.32^\circ$, $t = 2$

(f) Ply 6 : $\phi = -80.39^\circ$, $t = 2$

(g) Ply 7 : $\phi = -80.74^\circ$, $t = 1$

(h) Ply 8 : $\phi = -80.73^\circ$, $t = 1$

(i) Ply 9 : $\phi = -80.85^\circ$, $t = 1$

(j) Ply 10 : $\phi = -80.85^\circ$, $t = 2$

(k) Ply 11 : $\phi = -80.85^\circ$, $t = 2$

(l) Ply 12 : $\phi = -86.10^\circ$, $t = 1$
Fig. 6.8 do not allow for an intuitive interpretation. However, as already observed in [89] the reinforcement do not only concentrate on the mass region but also the right half of the structure. This is due to the fact that reinforcing the mass region will shift the first eigenmode to the yet weaker right half of the structure. The asymmetry with respect to the geometric symmetry axis may be explained by a kind of a self-excited instability: once an asymmetric reinforcement is applied, there is potentially no longer a pressure to remove this asymmetry. Further reinforcements will even be shifted to exploit it. Hence, the effect already observed in the single-ply experiment (Sec. 6.3.1), where a majority of the runs found an asymmetric patch geometry seems to be amplified in the here investigated thicker laminates.

### 6.4 Application example: Mass optimization of an airplane side rudder

The structure corresponds to the one presented in Sec. 5.5.1 and Fig. 5.7.

A composite structure shall be compared to a reference metal construction manufactured from aluminum. The fiber reinforced component shall be optimized for mass subject to constraints on strength and stiffness, i.e. the displacements at three reference points (A, B, and C in Fig. 6.4) shall be at most equal to the ones of the reference construction. The reference points
are positioned such that both bending and torsional displacements can be captured. The optimization problem is formulated as:

$$\min_x \mathcal{M}(x) \quad (6.15)$$

subject to:

$$\|\tilde{u}_A(x)\| - \|\tilde{u}_{A,ref}\| \leq 0 \quad (6.16)$$
$$\|\tilde{u}_B(x)\| - \|\tilde{u}_{B,ref}\| \leq 0 \quad (6.17)$$
$$\|\tilde{u}_C(x)\| - \|\tilde{u}_{C,ref}\| \leq 0 \quad (6.18)$$
$$\max_{\Omega} \{i_{\text{TsaiHill}}(x)\} - 1 \leq 0 \quad (6.19)$$

Where $\mathcal{M}$ denotes the structural mass and $\|\tilde{u}\|$ a displacement magnitude in the specified reference point. The Tsai-Hill failure criterion is applied to predict a first ply failure: The criterion computes a failure index $i_{\text{TsaiHill}}$ from the direction-wise stresses in each ply. It is evaluated at every layer top and bottom interface for each integration point for each finite element in the structure $\Omega$. Failure is predicted for $i_{\text{TsaiHill}} > 1$.

Scaling and an exterior penalty method (cf. Sec. 2.4.2) are used to transform this problem to an unconstrained, scalar minimization problem:

$$\min_x F(x) = \frac{\mathcal{M}(x)}{\mathcal{M}_{ref}} + p \left[ \max \left\{ 0, \frac{\|\tilde{u}_A(x)\| - \|\tilde{u}_{A,ref}\|}{u_s} \right\}^2 ight. \]
$$
$$+ \max \left\{ 0, \frac{\|\tilde{u}_B(x)\| - \|\tilde{u}_{B,ref}\|}{u_s} \right\}^2 
+ \max \left\{ 0, \frac{\|\tilde{u}_C(x)\| - \|\tilde{u}_{C,ref}\|}{u_s} \right\}^2 
+ \max \left\{ 0, \frac{\max_{\Omega} \{i_{\text{TsaiHill}}(x)\} - 1}{i_s} \right\}^2 \right] \quad (6.20)$$

The values of the metal construction are taken as a reference: These are $\mathcal{M}_{ref} = 4.08kg$, $\|\tilde{u}_{A,ref}\| = 0.6689mm$, $\|\tilde{u}_{B,ref}\| = 1.2938mm$, and $\|\tilde{u}_{C,ref}\| = 1.3665mm$. The scaling values are set to $u_s = 10^{-3}mm$ and $i_s = 10^{-3}$. The penalty factor is set to $p = 10$.

The internal structure on ribs, spar, and trailing edge are covered by laminates from carbon/epoxy prepreg material (material 1, Tab. B.1(a)): for the spar and the trailing edge $[\pm 30]_s$ (0.4mm thickness, zero-direction along the vertical axis) and $[\pm 45]_s$ for the ribs (0.4mm thickness). The laminate on the
6.4 Mass optimization of an airplane side rudder

Figure 6.10: The skin of the rudder is partitioned to 36 lamination sections. The laminate of the 18 sections illustrated here is mirrored at the \( x-z \)-plane to the second half-shell. The reference coordinate system \( \xi - \eta \) with angle \( \phi \) is used to determine the fiber direction.

The skin shall be optimized. It is initially covered by a ply of a very compliant, isotropic fill material (Young’s modulus \( E = 1MPa \), Poisson’s ration \( \nu = 0 \), specific mass \( \rho = 0kg/m^3 \), thickness 0.01mm). The skin is partitioned to 36 sections (Fig. 6.10). The section properties of the 18 starboard sections are mirrored at the \( x-y \)-plane onto the port side in order to obtain a structure with symmetric properties.

The skin reinforcement shall consist of either material 1 (Tab. B.1(a), ply thickness 0.125mm) or material 2 (Tab. B.1(b), ply thickness 0.1mm). The reinforcement plies are applied symmetrically on top and bottom surfaces of the laminate in order to obtain symmetric laminates. A maximum number of ten patches shall be applied. Each patch may have a thickness of at most ten times the raw material thickness.

Results and discussion: Convergence plots are given in Fig. 6.11. The best solution found has a mass \( M \) of 2.6040kg. The displacement values are \( \| \tilde{u}_A \| = 0.6523mm \), \( \| \tilde{u}_B \| = 1.2573mm \), and \( \| \tilde{u}_C \| = 1.32595mm \). The Tsai-Hill failure index \( i_{TsaiHill} \) reaches 0.125088. Hence, the design is in the feasible region. It is covered by five patches which are illustrated in Fig. 6.12.
Figure 6.11: Convergence plots for the side rudder application. Mass $\mathcal{M}$, displacements $\|\tilde{u}_A\|$, $\|\tilde{u}_B\|$, and $\|\tilde{u}_C\|$, Tsai-Hill failure index $i_{\text{TsaiHill}}$ and number of plies of the best solution in the pool as a function of the number of evaluations.
6.5 Concluding remarks

A method for the global optimization of locally varying laminates has been investigated. It builds on a genetic representation tailored to the requirements of laminate optimization problems. The method employs graph concepts in the encoding of the geometric region covered by a reinforcement treatment. However, abandoning a purely graph-based genotype like in existing methods ([93, 89, 133]) leads to more compact representations. The genetic search is assisted by gradient information in operations modifying the patch shape and determining optimal ply angles. The optimization method is applicable to a wide set of laminate optimization problems. Applications on eigenfrequency and stiffness- and strength-constrained mass optimization problems are presented.

The methods behavior is investigated in two academic example problems.
Whilst for the first experiment (Sec. 6.3.1) the results seem to be of reproducible quality, the more complex experiment (Sec. 6.3.2) can not confirm this assumption. However, also there, the method is able to find significantly improved designs in all runs.

The application to a side rudder structure illustrates a possible use-case of the method. There, an obvious partitioning of the structure into geometric sections is given by the construction. The method is able to find feasible solutions in this constrained problem.
Chapter 7

Genetic Encoding of Planar Spline Arrangements

This chapter investigates the genetic encoding of arrangements in TOD. A survey of topology optimization methods is given in Sec. 1.1.1. The here investigated method is dedicated to isotropic structures. An extension to laminated composites is discussed in Chap. 9.

7.1 Method

The method builds on a sequential EA as outlined in Alg. 1. A representation scheme, i.e. a genotype structure and associated genetic operators, based on geometric arrangements of spline curves is developed.

This section shortly discusses the problem-independent components of the algorithm. The representation and corresponding states (i.e. Alg. 1, state 1 and 6) are presented in detail in the following sections: the fitness is assembled by the separation method presented in Sec. 2.4.1. Rank-based stochastic universal sampling (SUS, [19]) is implemented as mating selection for the reason of minimal spread and bias (cf. [113]). A linear selection rate \( r(x) \) is computed for each individual:

\[
r(x, M) = \frac{\text{rank}^-(x, M) + \alpha}{\sum_{i \text{ size}(M)} i + \alpha}
\]  

(7.1)
rank$^-$ ($x, M$) denotes the rank of $x$ in the population $M$ ordered from worst to best. The tuning parameter $\alpha$ is used to influence the selection pressure and is set to 0.5 in this application. Environmental selection is done by comma selection with weak elitism: the offspring $M''$ replace the parents $M(t)$. If the best solution $^o x$ in $M''$ is worse than the best solution $^p x$ in $M(t)$ $^p x$ replaces the worst individual in the new parent population $M_{(t+1)}$. The algorithm is stopped if $t$ is larger than a predefined number or if there is no improvement in $t_s$ iterations.

7.1.1 Planar arrangements

In two-dimensional Euclidean space a set of curves $C$ induces a subdivision of the plane into two-dimensional, one-dimensional and zero-dimensional topological entities, called faces $F$, edges $E$, and vertices $V$ respectively. These entities and their topological relations are usually referred to as an arrangement $A(C)$ of the curves $C$. Thus, an arrangement describes a topological model induced by geometric curves as a triplet $(V, E, F)$, where $V$ is a finite set of vertices related to geometric points, $E$ is a binary relation on $V$, and $F$ is a relation in $E$. An edge $e$ associates two vertices $v_{(i,j)}$. A face associates none (in the case of the unbounded face) or one set of edges $e_o$ for its outer boundary and none, one or more sets of edges $e_i$ for inner boundaries (holes). Furthermore, an arrangement maintains a mapping from the curves $C$ to edges $e$ they induce, i.e. the arrangement history. Although arrangements are extensively investigated in literature on computational geometry and topology representation (e.g. [73, 6, 109] and references therein), they are seemingly not yet used as a representation concept in a structural optimization algorithm elsewhere.

7.1.2 A topology optimization problem formulated on arrangements

The arrangement concept is employed as a representation scheme for planar TOD problems (Fig. 7.1) of the form: Find an arrangement $A(C)$ induced by planar spline curves $C$ in the domain $\Omega$, and a set of face properties $F_p$ (from material, void) associated with the faces in $A(C)$ such that a criterion $F$ is minimized and constraints $h = 0$ and $g \leq 0$ are fulfilled. The domain $\Omega$ contains a set of required region $cR = \{cR_1, \ldots, cR_k\}$ associated with loads or boundary conditions. A legal design shall connect all these required regions
7.1 Method

\[ F_p \text{ such that all } cR \in cR \text{ are connected} \] (7.5)

7.1.3 Genetic encoding of planar arrangements

An arrangement maintains information on the topology, the shape and the history of a candidate solution. The topology is defined by the adjacency information of the entities in the arrangement. Two different graph representations of arrangements are common to represent this topology information (see Fig. 7.2): The adjacency of the faces can be interpreted from its face-graph (cf. Fig. 7.2(c)). The adjacency information of edges and vertices, as well as a common storage scheme, is given by a so called Doubly Connected Edge-List (DCEL) (Fig. 7.2(b)). The faces are associated with information on their structural attributes, i.e. void or the solid material. Edges are associated with curve segments resulting from intersecting all members of \( C \).

Illegal solutions are avoided in the design of the variation operators presented in the following. The face-graph is employed to evaluate the connectivity condition in (7.5): Therefore, a filtered view of the face-graph is constructed containing all faces with material properties and additionally the set of faces adjacent to required regions \( cR \). A connected component analysis on this filtered view then allows to directly identify disconnected individuals. The genotype investigated in this study groups genetic information on two different levels:
(a) The planar arrangement $\mathcal{A}(C)$ induced by three curves $C = \{c_0, c_1, c_2\}$ consists of six faces $F = \{f_0, f_1, \ldots, f_5\}$ ($f_0$ is unbounded), six vertices $V = \{v_0, v_1, \ldots, v_5\}$ and nine edges $E$.

(b) Doubly connected edge-list (DCEL) of the arrangement $\mathcal{A}(C)$. Each edge is represented by a pair of so called half-edges. These are directed and maintain the information on adjacency between vertices, edges, and faces.

(c) Face-graph of the arrangement $\mathcal{A}(C)$. This graph represents adjacency information of faces $F$.

Figure 7.2: Graph-representations in planar arrangements.

**Curve-Level:** In the here investigated examples, the type of curves in $C$ is restricted to interpolated cubic B-splines as they are usually implemented in commercial CAD and Computer Aided Engineering (CAE) packages. These are defined by a variable number of points $P$. The coefficients of the underlying piecewise polynomials are computed by interpolation in such a way that the resulting spline passes through all the points $P$. $P$ should not be confused with the control polygon of B-Splines (cf. [63]) which is usually not accessible in commercial software. Thus, the genotype contains a list of variable length representing $C$. A variable-length sequence of points is used to define each curve $C_i$ in $C$. Finally, the Cartesian coordinates of each point on each curve are represented by real-valued genes. Upper and lower limits may be specified for each gene in order to restrict the points $P_i$ to a design space $\Omega$. For convenience $\Omega$ is assumed to be a rectangle aligned with the two coordinate directions $x$ and $y$.

**Face-Level:** From the definition of an arrangement it is clear that the number and topology of the faces $F$ depends on the actual shape of the curves in $C$. Thus, given $C$ the face-graph of $\mathcal{A}(C)$ is well defined by a partition of $\Omega$. The resulting list of faces $F$ is associated with corresponding binary genes encoding the face properties $\text{void}$ or $\text{filled}$.

The two levels are not independent: The partition of $\Omega$ on the curve level leads to a fixed face-topology on the face-level. However, the mapping from the curve-level to the face level is not one-to-one: a certain face topology may
7.1 Method

originate from infinitely many configurations of C. If no restrictions on the number of curves or the number of points on each of the curves are considered, the representation is basically able to encode at least a very close approximation of any possible design in Ω. However, limiting the maximal number of points and curves tends to force smoother and more compact shapes.

The proposed genotype representation is of variable-length, i.e. the number of points on each spline as well as the number of spline curves in the arrangement (and so the number of faces) is not fixed but adapted during the optimization. Furthermore, if two adjacent faces share the same properties the separative edge in-between is not expressed in the corresponding structural model – such an edge is called non-phenotypic. This may even lead to so called non-phenotypic curves, i.e. curves inducing only non-phenotypic edges. Moreover, the same shape of a curve may be approximated by a different number of points. The effects of non-phenotypic regions in variable-length genotypes have been studied before and the so called bloat-effect has been described mainly in the field of genetic programming (cf. [200, 65, 22]). There, it can be observed that variable-length representations in EAs tend to accumulate non-phenotypic information during an optimization run, finally slowing down the convergence of the search. It can be argued that a certain amount of non-phenotypic information is useful in genetic variation where it may become phenotypic in the offspring again. However, there seems to be a trade-off in the amount of bloat appropriate in variation and the drawback in convergence it induces. Thus, measures to limit or even counteract bloat have to be considered when designing variation operators for this kind of genotype.

The fitness is evaluated by finite element models. The modeling requires a finite element mesh and therefore a re-meshing of the structure for each evaluation. Re-meshing is known to be a source of numerical noise which can be exploited by an optimization algorithm. Hence, solutions are only considered legal if a certain mesh quality (measured in terms of element distortions and size) can be achieved by an automatic mesher. In order to attain an acceptable mesh quality, non-phenotypic edges are ignored at this stage (cf. Chap. 8). Illegal solutions are filtered out of the pool before evaluation.

7.1.4 Genetic variation on arrangements

When creating a genetic encoding for EAs the states 1 and 6 in Alg. 1 have to be adapted.

**Initialization:** A random number of curves is created. In order for each individual to contain the required regions $\mathcal{R}$, random points inside these
regions are created. These points are then connected by random curves. Face properties are then assigned to the resulting faces. Legality of the individual is maintained, by ensuring a connected load path (i.e. a path of filled faces in the face-graph) between each region in $cR$.

In the following, the different genetic operators are presented. They are applied by chance, i.e. by user-defined application probabilities.

**Gaussian Curve Mutation:** Gaussian curve mutation is an adaption of the commonly used Gaussian mutation operator in real-encoded problems. It consists of adding a random vector from a Gaussian distribution to each point $P_j$ of a randomly selected curve $C_i$. The algorithm is outlined in Alg. 19. Points in constrained regions $cR$ are not modified in order to maintain a legal solution. Points moved outside of the design space $\Omega$ are reset to the closest location on the boundary of $\Omega$ (for the reasons of readability this mechanism is not explicitly described in Alg. 19). If the mutation does not result in topological changes of the face-graph, the face properties are set in a way to retain the face-graph of the parent individual. If new faces emerge, their properties are set at random. In this study the covariance matrix $\Sigma$ is fixed and isotropic $\Sigma = \sigma \cdot 1$ where $\sigma$ is a problem-dependent real value. Gaussian curve mutation is designed as a shape adaption operator. Therefore, the choice of curves is restricted to phenotypic ones and the choice of $\sigma$ has to be small compared to the dimensions of $\Omega$.

---

Algorithm 19 Gaussian Curve Mutation

Require: A parent individual $p_\mathbf{x}$

Require: A list of constrained regions $cR$

Ensure: An offspring individual $o_\mathbf{x}$

1: Copy parent $o_\mathbf{x} \leftarrow p_\mathbf{x}$
2: Collect a list of all curves $pC$ in $C$ of $o_\mathbf{x}$ inducing phenotypic segments
3: Randomly select a curve $pC_i$ out of $pC$
4: for each point $P_j$ in $pC_i$ do
5:     if $P_j$ is not in $cR$ then
6:         Replace $P_j \leftarrow P_j + \mathcal{N}(0, \Sigma)$
7:     end if
8: end for
9: Update face properties in $o_\mathbf{x}$
Insert Curve Mutation: Insert curve mutation is introduced in order to enable new faces to be generated by splitting existing ones. The algorithm is outlined in Alg. 20. After splitting a face, only one of the two resulting faces is set to filled.

Algorithm 20 Insert Curve Mutation

Require: A parent individual $p_x$
Ensure: An offspring individual $o_x$

1: Copy parent $o_x ← p_x$
2: Select a random face $F_i$ in $o_x$
3: Select to random edges $\{e_0, e_1\}$ on the outer boundary of $F_i$
4: for each edge $e_i$ in $\{e_0, e_1\}$ do
5: Create a random point $P_i$ on $e_i$
6: end for
7: Create a random spline from $P_0$ to $P_1$
8: Update face properties in $o_x$

Remove Curve Mutation: Remove curve mutation is created to reduce non-phenotypic curves and therefore counteracting bloat. It is outlined in Alg. 21. It removes all non-phenotypic curves and additionally one randomly selected curve. The face properties are set in a way to recover the face-graph of the parent individual if possible. New coinciding faces get properties at random.

Algorithm 21 Remove Curve Mutation

Require: A parent individual $p_x$
Ensure: An offspring individual $o_x$

1: Copy parent $o_x ← p_x$
2: for each curve $C_i$ in $C$ of $o_x$ do
3: Compute the phenotypic length $l(C_i)$
4: if $l(C_i) = 0$ then
5: Remove curve $C_i$
6: end if
7: end for
8: Randomly select a curve $C_j$ according to the rate $\sum_i l(C_i) / l(C_j)$
9: Remove $C_j$
10: Update face properties in $o_x$
Face Property Mutation: This mutation flips the property of a randomly selected face. The selection probability is inverse proportional to the area of the faces. Thus, small faces are selected more often for this mutation in order to keep the phenotypic changes low.

Relocation Mutation: Relocation mutation is introduced to redistribute material in the design space by shape changes, but keeping the overall volume approximately constant. The algorithm is outlined in Alg. 22. The basic idea is to introduce two random points: a source $P_{source}$ and a sink $P_{sink}$. Two different artificial displacement fields are then overlaid to get new positions for all the curve-points in the parent. The first field moves points towards the sink point $P_{sink}$, the other one moves them away from the source point $P_{source}$. The displacement magnitude each point takes depends on its distance to the respective point $P_{sink}$ or $P_{source}$. Face properties are updated in a way to keep the face-graph of the parent individual if possible. Random properties are assigned to new faces.

Algorithm 22 Relocation Mutation

Require: A parent individual $p_x$

Require: A list of constrained regions $c_R$

Ensure: An offspring individual $o_x$

1: Copy parent $o_x \leftarrow p_x$
2: $r \leftarrow$ random real in $[r_{min}, r_{max}]$
3: Create a random point $P_{sink}$ inside the boundary of $o_x$
4: Create a random point $P_{source}$ inside the boundary of $o_x$
5: for each point $P_i$ in all curves $C_j$ of $o_x$ do
6: if $P_i$ is not in $c_R$ then
7: $P'_{sink,i} \leftarrow P_{sink} + \frac{\sqrt{||P_i - P_{sink}||^2 + r^2}}{||P_i - P_{sink}||} (P_i - P_{sink})$
8: $P'_{source,i} \leftarrow P_{source} + \frac{\sqrt{||P_i - P_{source}||^2 + r^2}}{||P_i - P_{source}||} (P_i - P_{source})$
9: $P_i \leftarrow \left( P'_{sink,i} + P'_{source,i} \right) / 2$
10: end if
11: end for
12: Update face properties in $o_x$

Curve Merge Mutation: This mutation operator is designed to reduce bloat (Alg. 23). It recovers the boundary of a parent solution and approximates it by creating a new set of curves. The number of curves is set to the
minimum required to induce approximately the same boundary. The number of points on each curve is set randomly. Small features are potentially neglected.

Algorithm 23 Curve Merge Mutation

Require: A parent individual $p_x$

Require: A list of constrained regions $c_R$

Ensure: An offspring individual $o_x$

1: Initialize an empty individual $o_x$
2: Select all phenotypic edges $p_e$ in $p_x$
3: for each edge $e_i$ in $p_e$ do
4: Create a random ordered sequence of points $P_i$ on $e_i$. Locations of high curvature on $e_i$ get a higher probability for a new point.
5: for each point sequence $P_j$ if $j < i$ do
6: if one endpoint $Q$ of $P_j$ coincides with one endpoint in $P_i$ and $Q$ is not in $c_R$ then
7: append $P_i$ to $P_j$
8: end if
9: end for
10: end for
11: for all point sequences $P_i$ do
12: Create a curve $C_i$ from $P_i$ in $o_x$
13: end for
14: Update face properties in $o_x$

Insert Bubble Mutation: Insert bubble mutation is inspired by the Bubble Method (cf. [75]). It introduced topological changes by creating a void region – i.e. a bubble – with random shape at a random position inside the boundaries of a parent individual. The algorithm described in Alg. 24 keeps the overall volume approximately constant by inflating the contour of the parent individual before actually creating the bubble. The property of the newly created bubble is set to void, whilst the other faces are set according to the ones in the parent individual. Although the points defining the boundary of the bubble are on a circle the resulting bubble-boundary does not necessarily approximate a circle but depends on their number.

Split-and-Splice Curve Crossover: Split-and-splice curve crossover is a generalization of the one-point crossover operator found in literature (cf.
Algorithm 24 Insert Bubble Mutation

**Require:** A parent individual $p_x$

**Require:** A list of constrained regions $c_R$

**Ensure:** An offspring individual $o_x$

1: Copy parent $o_x \leftarrow p_x$
2: $r \leftarrow$ random real in $[r_{min}, r_{max}]$
3: Create a random point $P_{center}$ inside the boundary of $o_x$
4: **for** each point $P_i$ in all curves $C_j$ of $o_x$ **do**
5:     **if** $P_i$ is not in $c_R$ **then**
6:         $P_i \leftarrow P_{center} + \sqrt{\frac{\|P_i - P_{center}\|^2 + r^2}{\|P_i - P_{center}\|}} (P_{center} - P_i)$
7:     **end if**
8: **end for**
9: $n \leftarrow$ random integer in $[4, n_{max}]$
10: Create a circular ring of $n$ random points $P_{bubble,i}$ with radius $r$ from center $P_{center}$
11: Add a new curve $C_i$ defined by $P_{bubble,i}$ to $o_x$
12: Update face properties in $o_x$

Chap. 3). It takes two parent individuals and creates two offspring. It splits the sequence of curves in each parent at a random position and recombines the resulting head and tail sequences. Since the curve sequence is of variable length, a minimal and maximal allowed number of curves has to be respected in the creation of the splitting points. The face properties of the offspring are then set according to the properties in the parents by location each face of offspring $o_1 x$ in parent $p_1 x$ and taking its properties, and the same for $o_2 x$ and $p_2 x$.

**Face Property Crossover:** Face property crossover exchanges the face properties at random locations between two candidate solutions. The algorithm is outlined in Alg. 25.

**Overlay Crossover:** Overlay crossover is based on overlaying the arrangement of two parent solutions and reassembling their face properties. The algorithm is given in Alg. 26. The face properties are interpreted as logicals (filled as TRUE and void as FALSE). The overlay of the face properties is then based on the logical operations $\text{AND} (\wedge)$ and $\text{OR} (\lor)$. 
Algorithm 25 Face Property Crossover

Require: Two parent individuals \( p_1^x, p_2^x \)
Ensure: Two offspring individuals \( o_1^x, o_2^x \)

1: Create offspring \( o_1^x \leftarrow p_1^x \)
2: Create offspring \( o_2^x \leftarrow p_2^x \)
3: Set face properties in \( o_1^x \) and \( o_2^x \) to \textit{void}
4: for each face \( F_{i,1} \) in \( p_2^x \) do
5: \( P_{F,i} \leftarrow \) arbitrary point in \( F_{i,1} \)
6: Locate face \( \hat{F}_{i,0} \) at \( P_{F,i} \) in \( o_1^x \)
7: Set property of \( \hat{F}_{i,0} \) to property of \( F_{i,1} \)
8: end for
9: for each face \( F_{i,0} \) in \( p_1^x \) do
10: \( P_{F,i} \leftarrow \) arbitrary point in \( F_{i,0} \)
11: Locate face \( \hat{F}_{i,1} \) at \( P_{F,i} \) in \( o_2^x \)
12: Set property of \( \hat{F}_{i,1} \) to property of \( F_{i,0} \)
13: end for

Algorithm 26 Overlay Crossover

Require: Two parent individuals \( p_1^x, p_2^x \)
Ensure: Two offspring individuals \( o_1^x, o_2^x \)

1: \( C \leftarrow \) all curves in \( \{p_1^x, p_2^x\} \)
2: Create offspring \( o_1^x \) from \( C \)
3: Create offspring \( o_2^x \) from \( C \)
4: for each face \( F_{i,0}, F_{i,1} \) in \( o_1^x, o_2^x \) do
5: \( P_{F,i} \leftarrow \) arbitrary point in \( F_{i,0} \)
6: \( p_{i,0} \leftarrow \) face property at location \( P_{F,i} \) in \( p_1^x \)
7: \( p_{i,1} \leftarrow \) face property at location \( P_{F,i} \) in \( p_2^x \)
8: Set face property of \( F_{i,0} \) to \( (p_{i,0} \land p_{i,1}) \)
9: Set face property of \( F_{i,1} \) to \( (p_{i,0} \lor p_{i,1}) \)
10: end for
**Geometric Crossover:** Geometric crossover splits two parent individuals at a random straight line and recombines the resulting parts in two offspring solutions (Alg. 27). In order to counteract bloat the boundary of the offspring solutions is later approximated by a new set of curves. The approximation procedure is the same as the one described in the *Curve Merge Mutation*.

**Algorithm 27 Geometric Crossover**

**Require:** Two parent individuals $p_1^x, p_2^x$

**Ensure:** Two offspring individuals $o_1^x, o_2^x$

1: Create a random straight line $L$ in $\Omega$
2: Split all curves $p_1^x$ at $L \rightarrow \{-C_0, +C_0\}$
3: Split all curves $p_2^x$ at $L \rightarrow \{-C_1, +C_1\}$
4: Create offspring $o_1^x$ from $\{-C_0, +C_1, L\}$
5: Create offspring $o_2^x$ from $\{-C_1, +C_0, L\}$
6: Update face properties in $o_1^x$
7: Update face properties in $o_2^x$
8: Approximate the boundary of $o_1^x$ by a new set of curves $C_0$
9: Approximate the boundary of $o_2^x$ by a new set of curves $C_1$

**7.2 Implementation**

The method is implemented in the Python programming language (version 2.4 and 2.5). Geometry operations, meshing, and finite element analysis are performed in the commercial software ABAQUS CAE and ABAQUS Standard (version 6.8). Evaluation is done in parallel by employing Grid Engine 6 N1 on a 28 node compute cluster.

**7.3 Application example: Minimal compliance cantilever design**

The minimal compliance cantilever problem is a commonly used showcase application for topological optimum design. Minimal compliance design is a widely investigated topic and can efficiently be tackled by mathematical programming (and hence homogenization approaches, see Sec. 7.4). The domain is given by a rectangular region depicted in Fig. 7.3. Two support points on the left and a load introduction point at the right side are defined. The
goal is to find the stiffest structure for a given amount of material. Here, the optimization problem is directly defined as:

\[
\begin{align*}
\text{minimize } & \quad u_{\max}(x) \\
\text{subject to } & \quad M(x) - M_{\text{adm}} \leq 0
\end{align*}
\]

where \(u_{\max}(x)\) is the maximal displacement (magnitude) at any point in the structure, \(M(x)\) is the mass of the structure, and \(M_{\text{adm}}\) is the maximum admissible mass \((M_{\text{adm}} = 20.25\,\text{gr} \text{ or } 30\% \text{ of } \Omega)\). A linear elastic structural analysis is carried out to find the maximal displacement \(u_{\max}\). A material with Young’s modulus \(E = 70\,\text{GPa}\) and Poisson’s ratio \(\nu = 0.3\) is employed.

The presented results correspond to the best solution found in several experiments with different settings. The optimization is run over 400 generations with a population size of 250 individuals \((100000 \text{ evaluations})\). The application rates of the operators are set to: \(15.625\%\) for overlay crossover and Gaussian curve mutation, \(12.5\%\) for split-and-splice crossover, \(9.375\%\) for curve merge mutation, \(6.25\%\) for relocation mutation, insert bubble mutation, and face attribute crossover, and \(3.125\%\) for face attribute mutation, insert curve mutation, and remove curve mutation each. The maximum number of curves is limited to 12 with a maximum of 12 points on each. This results in a maximal number of 288 real-valued genes encoding coordinates and additionally an undetermined number of boolean genes encoding face properties.
Figure 7.4: Best solution found for the minimal compliance cantilever problem.

The fittest solution is drawn in Fig. 7.4. Convergence plots are given in Fig. 7.5. The maximum displacement of this solution is \( u_{\text{max}} = 2.51 \times 10^{-3} \text{mm/N} \). It has a mass of \( 20.23 \times 10^{-3} \text{kg} \). Its shape is defined by 11 curves given by 110 points.

The shape of the best solution shows two connected load paths between load introduction and the two support points. These carry tension and compression stresses induced by bending. The region in-between is characterized by a group of seven wholes cutting the shear field into six bridges. The convergence plots in Fig. 7.5 indicate a stagnation after 200 generations.

A tendency to accumulate bloat can be seen in Fig. 7.5(c) and 7.5(d). The overall amount of genetic information in the representation increases with each generation. This increase is partly motivated by more complex shapes in subsequent iterations.

### 7.4 Comparison with a Homogenization Model

The solution, shown in Fig. 7.4 and obtained with the presented method, is compared with that of a voxel-based homogenization method invented by [36]. It idealizes the design space with a regularly spaced finite-element mesh to identify a topology by assigning to the elements densities with unity or close-to zero values. The result depends upon the design space geometry, the kinematic and kinetic boundary conditions, a specified average mass density, and mesh properties. The latter includes the fineness of the mesh and char-
Figure 7.5: Convergence plots for the cantilever application. Displacement (a), mass (b), number of points (c), and number of curves (d) over generations $t$. The solid line indicates the value of the best individual in the population, the dashed line corresponds to the population average.
Figure 7.6: Optimum design of the cantilever problem found by a homogenization method
characteristics of the employed finite elements themselves. The original method minimizes the work of the external forces,

\[ f(\rho) = W(\rho) = u^T r \]  

hence the name minimum compliance problem. The problem has been shown to be convex under the constraint of constant average density, or total mass,

\[ h(\rho) = \sum_{k=1}^{N_{el}} \rho_k \cdot V_k - \overline{\rho} \cdot V \]  

The inherently discrete problem is transformed into a smooth one by introducing a density function on each finite element. In case of isotropy the densities are connected with Young’s modulus by

\[ E(\rho) = E_0 \rho^p, \quad \rho_{\text{min}} < \rho < 1, \]  

where the exponent is often chosen \( p = 4 \) and bounding \( \rho \) with a small minimum keeps the system matrices regular. Our implementation of the method is based on minimizing the Lagrangian

\[
L(\rho, \Lambda, \lambda^+_{i}, \lambda^-_{i}) = W(\rho) \\
+ \Lambda \left( \sum_{i=1}^{N_{el}} \rho_i V_i - \overline{\rho} V \right) \\
+ \sum_{i=1}^{N_{el}} \lambda^+_{i} (\rho_i - 1) \\
+ \sum_{i=1}^{N_{el}} \lambda^-_{i} (\rho_{\text{min}} - \rho_i)
\]  

We use elements with bi-linear approximation functions and repress checker board solutions with applying a blur filter algorithm to the gradient vector. Sequential topology improvements are found with Cauchy’s method and the step width is determined so that the next candidate finite element reaches unit density. This determines the number of iterations:

\[ N_{\text{iter}} = \rho N_{el} \]  

At the end of each line search a number of elements may have received densities violating the lower bound which problem is mitigated by assigning the
minimum densities, respectively, and a subsequent correction of the total mass balance. The number of iterations can be significantly reduced below (7.12) by replacing in each of them all element densities, having a specified percentage less than unity, with unity. The in Fig. 7.3 specified problem is here solved with a mesh of 250 × 100 square-shaped finite elements. A tolerance level of 10% reduces the number of iterations to convergence to \( N_{\text{iter}} = 102 \). The filter averages the values of the respective reference elements and their adjacent eight neighbours. The maximum normalized displacement of the optimal design found by this method, and shown in Fig. 7.4, is \( 2.17 \cdot 10^{-3} \text{mm}/N \). This corresponds to an improvement of 15.6% over the solution of the evolutionary approach.

The main differences can be found in the filigree truss structure obtained by the homogenization method and in its contour which utilizes the geometrical design space height to achieve maximum bending stiffness. Both features are apparently harder to achieve by the here proposed genetic encoding: Delicate truss structures require two boundaries on both sides of each bar-like member in order to model their geometry. Shape changes on these boundaries are especially sensitive as they tend to induce illegal solutions by either disconnecting load paths or inducing mesh distortions. Further on, the parametric requirements of such boundaries are higher than those of more compact shapes. Once a feasible topology is found, successful changes of its outer contour require the variation of more than one geometric item. This becomes evident from the observed behavior in the mass constraint (Fig. 7.5(b)). The margin for feasible variations rapidly decreases.

The homogenized model requires 25000 decision variables. The genetic representation evolves a design with only 222 genes. However, the HDM approach profits from the affordable derivative computation and the low dependencies between the decision variables.

### 7.5 Conclusions

A new representation concept for planar topological optimum design is presented. The method operates on a geometric model of the boundary and the corresponding topology it induces (called arrangement). The proposed genotype is of variable length. It consists of spline curves represented by a variable number of points on a first level (curve-level) and a variable but dependent set of faces associated with properties (filled or empty) representing the material in the corresponding geometric region. The exact structure of the genotype and therewith the dimensionality of the search space is adapted during the
optimization run. Initialization and variation operators of an EA are tailored to directly operate on the new representation.

Since the arrangement abstraction contains information on the shape and topology of a candidate solution, it enables for several geometric analysis operations at relatively low computational cost: The legality of an individual, i.e. if there is a connected path between load introduction and support, can be checked on an abstract geometric level. Furthermore, since a geometric model is available, the volume and therewith the mass, as well as the length of boundaries may be computed directly from the genotype. No extensive post-processing is required for the interpretation of the resulting shape. This, since it consists of geometric information which is widely compatible with common CAD- and CAE-environments.

The kind of representation concept is not limited to a certain structural domain or simulation method. Although, only finite element models for structural mechanics are considered within this study, one may think of any other kind of evaluation procedure requiring information on the boundary and topology of a candidate solution.

Several issues emerge from the investigated new representation offering interesting topics for further research:

The overall performance of the algorithm and especially the convergence once a potentially good topology is identified is not competitive to state-of-the-art homogenization approaches. Due to the large number of design evaluations, it is not appropriate for the use in a production environment. With increasing number of populations, changes in the topology of the best solution become improbable since they would often either violate constraints or degrade the mechanical properties of the best solution found so far. Further on, only shape variation conserving the topology can be observed. Genetic drift leads to an assimilation of the individuals in the population. A risk for premature convergence in a local optimum arises. Hence, after an initial number of generations with large improvements the convergence slows down until at a certain stage the improvements achieved are at a very low level – the fine-tuning of the shape is slow because only part of the variation operators are appropriate. Several improvements of the method could be derived from this observation: The application rates of variation operators need not to be constant but could be adapted on the run based on the current state of the optimization (cf. e.g. [61, 247, 92, 257]). Genetic drift or diminution of the genetic diversity in a population is especially critical if the population size is small compared to the dimensionality of the search space. Hence, methods reducing bloat and therewith the dimensionality of the search space, as well as methods increasing the population diversity could help to improve
the performance. Various efforts have been made in both directions (e.g. [65, 156, 15, 188]). Finally, hybridization of the EA scheme with methods from the field of mathematical programming or local heuristics like in [52] could help to improve the performance. Unlike in parametric genetic optimization, the variation operators include costly shape operations. A parallelization of not only the evaluation state, but also the variation state of the EA could help to reduce runtime.

A set of surfaces induces a partition of a three-dimensional space into vertices, edges, faces, and cells. Surfaces could be parameterized by a variable set of points or curves. Thus, a similar genetic representation could be employed to extend the method to three-dimensional TOD.

The following chapter (Chap. 8) discusses robustness issues in the context of geometric encodings like the one presented here. Chap. 9 presents extensions to laminated composites and three-dimensional shape changes.
Chapter 8

On the Robustness of Geometric Operations and Automated Meshing

The investigation of planar spline arrangements in Chap. 7 revealed challenges concerned with the robustness of geometric operations: it seems to be an inevitable fact that variation operations based on geometric features conditionally fail (cf. e.g. [257]). Thus, a considerable portion of the offspring may be illegal due to geometric defects caused by an unsuccessful mapping. This chapter aims at investigating sources and possible countermeasures for non-robust geometric operations.

8.1 Sources of geometric defects

The transformation of parametric splines $C$ to an arrangement $A$, i.e. the computation of spline segments forming edges, the computation of vertices marking start- and end-points on these segments, and the detection of faces requires the evaluation of geometric predicates [73]. Such predicates are for example a check if two points are coincident, if a point is on a curve segment, if two curve segments intersect, or if a curve segment is vertical etc. These predicates are bound to a so-called geometric kernel $K$, i.e. an algebraic structure defining the representation of basic geometric objects. A cartesian kernel for instance represents a point in affine $d$-dimensional Euclidean space by a set of $d$ coordinates. If no projective operation is required, a kernel may
be realized by ring operations only, i.e. addition and multiplication. If a division operation is required, e.g. by rational splines, this may be achieved by either using a homogeneous kernel or by extending the kernel to apply field operations. A homogeneous kernel represents a \( d \)-dimensional point by \( d + 1 \) coordinates, with the last coordinate as a common denominator. Field operations comprise addition, multiplication, and division.

In commercial CAE-applications and also in the here employed ABAQUS CAE software cartesian kernels with field operations are commonly employed [1]. Ideally the field numbers comprise all real numbers in \( \mathbb{R} \). However, on real computers, the operations are restricted to a subset of \( \mathbb{R} \), i.e. the so-called machine numbers \( \mathbb{M} \) with limited precision. Hence, there is a difference between the theoretical foundations of the geometric operations employed and their implementation in software, i.e. operations generating results not representable in \( \mathbb{M} \) are approximated in the implementation. The difference between the exact value and the computer approximation is usually called round-off error (the term is misleading, since it results from a truncation and not rounding operation in a strict sense). Therefore, kernel operations in applications based on machine numbers can only be approximate. As a consequence the evaluation of the above introduced predicates may return ambiguous results. Especially error-prone are predicates evaluating coincidence and equivalence. This is avoided by accepting a certain geometric tolerance \( \epsilon \) in these predicates, i.e. if the distance between two points is smaller than \( \epsilon \), these points are treated as coincident. In ABAQUS CAE this tolerance is set to \( 1 \cdot 10^{-6} \) by default [1].

Splines are piece-wise polynomial functions represented as parametric curves: without loss of generality we can assume the following spline representation [63, 199, 213]

\[
C = C(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix}, \quad t \in \mathbb{R}, \quad 0 \leq t < 1
\]  

(8.1)

In order to compute an arrangement \( A \), the set of splines \( C \) has to be partitioned into segments at their intersection points. The computation of these intersection points is usually done in the parametric domain of the curves, i.e. to compute the intersection of the splines \( C_1(t_1) \) and \( C_2(t_2) \) one seeks for parameters \( t_1^* \) and \( t_2^* \) where

\[
C_1(t_1^*) = C_2(t_2^*). \tag{8.2}
\]

Due to the polynomial character of splines, this corresponds to the search for roots of a polynomial of degree \( p \), i.e. the degree of the splines. A closed
form solution is only available for $p \leq 3$. Hence, usually approximative solutions have to be accepted. Moreover, equality comparison in Eq. 8.2 in the framework of an approximative kernel is affected by the geometric tolerance $\epsilon$. Therefore, the parameters $t_1^*$ and $t_2^*$ can only be determined to lie each in a finite interval $t_{1,l}^* \leq t_1^* \leq t_{1,h}^*$, and $t_{2,l}^* \leq t_2^* \leq t_{2,h}^*$ respectively. The length of this interval depends on the shape of the curves $C_1(t)$ and $C_2(t)$. The distance between $C_1(t_1^*,l)$ and $C_2(t_1^*,h)$ (and the same for the respective points on $C_2$) may be considerably larger than the geometric tolerance $\epsilon$ (Fig. 8.1). Even though, the behavior of the above naive method can be improved by several tricks, parametric geometries like splines are able to inflate inaccuracies from a level below a threshold $\epsilon$ up to an undesirable scale.

8.2 Improving the robustness of geometric operations

8.2.1 Escaping to exact geometry

Failure of geometric operations is intrinsic in approximate computation. Hence, a complete elimination of these failures is only achieved by employing exact computation. On a real computer exact computation is only possible by restricting operations to integer arithmetic. In the context of geometry, this may either mean the application of a kernel based on integer numbers $\mathbb{Z}$
and ring operations on these, or a kernel based on rational numbers $\mathbb{Q}$ and field operations thereon (since rationals can be decomposed into integer nominator and denominator). As long as all geometric objects are restricted to be representable in integers or rationals, these kernels would ideally provide no source of error except for overflow if the domain of machine representable integers is left (which is also the source of underflow for rationals). The range of integers representable on a computer depends on the number of bits provided for its storage, thus essentially determined by the chosen type in the implementation. Two experiments using exact computation to determine the topology of arrangements have been carried out:

**Approximation by polylines:** So-called polylines of straight line-segments are representable in an exact kernel of rationals. If all points on a polyline have rational coordinates, all intersection points of these polylines have obviously rational coordinates as well. This, since intersection points of straight lines can be determined by solving a linear system of equations. The above analyzed intersection of splines however leaves the space of rational numbers: Roots of polynomials (even though they have only rational coefficients) are real numbers. Thus, they are not representable in an exact kernel based on rationals. But, by sampling splines with polylines (Fig. 8.2) before topologically sweeping the arrangement, the geometric operations can be done by exact computation. If the curve parameters $t$ for the sampling points, the knots, and the control points of the splines are restricted to rational numbers, the sampling points will have rational coordinates as well. This

![Approximation of a spline by a polyline](image)

Figure 8.2: Approximation of a spline by a polyline
becomes obvious when remembering the splines’ polynomial properties involving only field operations. The number of sampling points (or the length of the straight line segments) determines the degree of approximation in the geometry as well as the overall number of geometric entities used when sweeping the arrangement. The approach has been implemented in the C++-programming language based on a framework provided by [256]. The method requires a large number of geometric comparisons to be evaluated during the generation of a topological representation. This number rapidly increases if a more accurate sampling of the curves is desired. Moreover, comparisons of rationals are computationally much more intensive than comparison of floating point numbers. This is mainly due to the fact that common desktop computers are equipped with a numerical co-processor tuned and dedicated to floating point operations solely. The computation of an arrangement by this approach turned out to be unacceptably slow.

Hybrid algebraic/rational representation: This approach employs two kernels. A rational kernel for the representation and operations on all but the intersection points, and an algebraic kernel to perform operations just on intersection points. The algebraic representation of polynomial roots is organized in a two step approach: In order to compare two algebraic numbers their provenance has to be known. Equality can be determined by checking if two algebraic numbers refer to the same root. For the comparison with rational numbers, algebraic numbers are associated with a bounding interval representable in rationals. If this interval is too large to allow for unambiguous comparisons, the stored origin of the respective algebraic number can be taken to refine the approximation. Every real number can be bracketed by a finite, but arbitrarily short interval of rationals. Thus, the quality of approximation is only limited by the implementation (i.e. over/underflow). The implementation is done in the same framework as the first approach. An implementation of algebraic number types is taken from [132]. Although requiring less comparison operations, now, more computation is required in determining intersection points by algebraic computations. The computation time required in these algebraic operations turned out to undergo unpredictable fluctuations, i.e. even for the same number of input data there can be orders of magnitude in-between the evaluation time of the fastest and slowest operation. Additionally, algebraic numbers cannot be saved to disk but have to be recomputed on loading inducing further difficulties in networked environments with heterogeneous platforms. Thus, the overall performance could not be improved over the first approach.
8.2.2 Shape healing and repair algorithms

A shape healing method accepts a possible failure of approximate geometric operations which allows to employ an inexact cartesian kernel. Instead of avoiding failures the results are checked for defects and a repair algorithm is employed in order to fix the cause or the undesired effects of a geometric failure. A particularly critical type of defect is an inconsistent mapping from geometric objects (like points, curves, surfaces, and volumes) to topological entities (vertices, edges, faces, and cells). A failed mapping leads to wrong connections in the model topology. These may be visualized as warped faces with zero or negative area for instance. Faulty associations may be detected by checking the topology and the inducing geometry by complementary construction principles: e.g. if an association has been constructed by the use of a left-of predicate, the result can be checked for consistency by employing a right-of and coincident predicate on the same objects in reversed order. Moreover, the boundary representation of the result may reveal topological defects in the form of inconsistent directions of half-edges for instance. However, in commercial applications, these low level operations are usually not exposed to the user interface. Hence, the detection of geometric failures has to be done on a trial-and-error base, i.e. by relying on the application internal geometry checks usually raising exceptions in the creation of a faulty feature. Once a defect is detected, a repair algorithm is employed to fix the problem. The repair may engage on the level of input data for the geometric operations or on the level of the generated result. The first approach requires more information on the source of failure whilst the second accepts an inconsistency between geometric input data and resulting topology. However, this inconsistency is no longer affected by machine specific settings but deterministic and reproducible as long as the employed algorithms are deterministic.

A shape healing approach is realized within the commercial ABAQUS CAE software: Geometric defects are detected by catching exceptions generated during the model creation and by employing rudimentary geometric checks available in the scripting interface of the software [1]. Once a defect is detected, a repair algorithm directly operating on the geometric input-data in the genotype representation is employed. The algorithm resamples and smooths spline curves in order to attain a simplified shape. Although the method is not able to repair all pathological configurations, it reaches a sufficient level of successful mappings at acceptable numerical cost.
8.3 Automatic meshing

The evaluation of a candidate solution requires a numerical model to predict its mechanical behavior. When FEM shall be employed, this requires a meshing of the domain of the structure, i.e. a partition into finite elements. The reliability of simulation results obtained by FEM depends on the geometric quality of this finite element mesh. In the context of structural optimization human interaction in the evaluation of potentially thousands of candidate solutions is clearly undesired. Hence, finite element meshes are created by automatic procedures. An optimization procedure operating on simulated mechanical qualities is not able to differentiate between meshing and shape variations. Thus, it tends to exploit the weaknesses of automated meshing procedures to improve the supposed numerical simulation results by degrading the mesh and hence the prediction quality.

This section aims at investigating sources of numerical errors and methods to reduce or control these errors. The investigations are carried out on an abstract level which mainly considers the geometric properties of the mesh, i.e. the actual element type and its implementation are not discussed. Then, general sources of errors can be found in the discretization and the numerical integration over the domain [267].

Discretization errors have to be expected if the true solution is not representable in the solution space of a finite element model. This is the case for most engineering applications. The set of representable solutions is determined by the number of degrees of freedom in the model and the element types employed. If we assume that the available element types do not neglect mechanical aspects relevant for a particular situation, then refining the element size or providing elements with higher-order shape functions – i.e. increasing the number of degrees of freedom in a model – will in general lead to better approximations. The same applies locally: If the geometry or loadings provoke large gradients in the displacement field a local refinement of the mesh can significantly improve the approximation quality by reducing discretization errors. As a disadvantage, refining the mesh increases the numerical requirements to obtain a solution. Discretization errors may show an orientation dependence (even for isotropic material models) if the finite elements have a large aspect ratio and are oriented in the same direction.

Integration errors are connected to element distortions: Elements employed in an undeformed configuration which shows angular deviations when compared to their master configuration are a source of integration errors.

Mesh generation has to consider these sources of errors. A control of the amount of the errors is possible by either controlling possible sources of errors.
or by detecting errors and optimizing meshes in iterative refinement steps based on several simulations.

The framework presented in Chap. 7 establishes minimal requirements on the mesh quality by a two level strategy: Possible sources of errors are avoided before meshing and the resulting mesh is checked by a set of quality criteria before evaluating the model. If these checks detect an insufficient quality the solution is not accepted as a legal solution and rejected. If the input geometry has small angles, the finite elements in these regions will inevitably have small angles and thus potentially large distortions as well. Internal edges (so called non-phenotypic edges) turned out to be a particularly critical source of errors (Fig. 8.3), hence these edges are ignored in the meshing process. A single-material structure is not influenced by this approach. Since the curves inducing internal edges are not removed from the genotype, the possibilities for further genetic modifications are not affected. Mesh validation is performed to rate shape and size metrics. Threshold values for the checks are problem specific input data and part of the optimization application setup. This approach is not able to completely eliminate meshing as a source of numerical noise: It sets a lower bound on the mesh quality by eliminating individuals where automated meshing with predefined input data is not able to achieve minimal requirements. In other terms, it narrows the search to solutions which can be meshed with sufficient quality. Thus, the optimization result depends on the input data for the automatic mesher and for the mesh check procedure. One has to take the poorest mesh quality into account when configuring an optimization application.

### 8.3.1 Projection method

Particularly critical for further investigations with more advanced face properties is the approach to mesh each face separately. There, one has to expect an increased number of phenotypic edges, since internal boundaries may separate sections with different properties within the same structure. Whilst in single-material structures (Chap. 7) an inappropriate influence on the mesh quality can be avoided by ignoring non-phenotypic edges, this approach is no longer appropriate for laminate sections.

Thus, for further investigations a method to map geometric objects onto a regular finite element mesh is developed (Fig. 8.3(c)). The method reduces element distortions to a bare minimum. Discretization errors due to a stair-like approximation of an initially curved shape are evident. However, since the mesh quality is no longer directly dependent on the geometric shape of the candidate solution, there is no bias and hence no possibility to exploit
these errors for an automated optimization procedure. Discretization errors have to be considered when evaluating local criteria like for example stress or strain-based failure criteria.

8.4 Concluding remarks

Inexact computation in geometric predicates is identified as a source for non-robust geometric operations. A complete elimination of this source would only be possible by employing exact computation at the expense of a huge increase in computation time. A method based on shape healing and repair mechanisms is employed instead. This method is a compromise between an increase in computational cost in all geometric variation operations and a reduced rate of failure.

In order to eliminate the risk to optimize the mesh instead of the structure the mesh quality of each model is validated before evaluation. Individuals with insufficient mesh quality are excluded from the evaluation. The approach to ignore non-phenotypic edges becomes prohibitive if a potentially large portion of internal edges becomes phenotypic in multi-material applications. Hence, for further investigations of geometry-based variation operators an unbiased meshing strategy based on projections is employed.
Chapter 9

Arrangement-based Laminate Optimization on Structures with Variable Shape

This chapter fuses the laminate optimization approach presented in Chap. 6 with the shape representation of Chap. 7. It aims at investigating possibilities to concurrently optimize the shape, topology, and laminate of a structure. A survey of combined optimization methods is given in Sec. 1.1.3.

9.1 Method

The here proposed method aims at extending the scope of the laminate optimization approach presented in Chap. 6 by a variable partitioning of the structure into laminate sections and a by allowing for shape variations of the structure.

Therefore, an asynchronous EA scheme is adapted to operate on a hierarchical representation encoding shape and laminate attributes at the same time. The EA scheme corresponds to Alg. 8 already employed in Chap. 6. The following sections detail the representation and genetic variation operators. The remaining algorithm components remain unchanged.
9.1.1 Genetic representation

The here investigated hierarchical representation is illustrated in Fig. 9.1. It comprises four levels: an arrangement level, a shape level, a laminate level, and a ply level. The arrangement level induces the partitioning of the structure into laminate sections. It is represented by a variable-length container of planar curves \( C \). Each curve is itself encoded by variable-length sequence of points (cf. Chap. 7). These curve induce a partitioning of a plane into faces of variable shape. Adjacency information between these faces is utilized in laminate variation operators by employing the concepts developed in Chap. 6.

The shape is described by a prototype geometry and a set of shape parameters \( \mathbf{s} x = \{s x_1, s x_2, \ldots, s x_n\} \). The partitions thereon are generated from the geometry on the arrangement level. There are two possibilities to achieve this: A translation of the arrangement to native surface coordinates on or a projection of the arrangement geometry onto the variable shape. The here investigated problems employ the later approach. Notably, the projection method is more restrictive what concerns feasible shapes. The laminate representation on the laminate level follows the patch concept already employed in Chap. 6. It uses the partitions on the shape level as laminate sections. The order of plies in the laminate representation determines the stacking sequence. The ply level finally represents the properties of single plies: It is associated to genes describing a ply angle, a ply thickness, and a ply material. Whilst continuous values are taken for ply angles \( \phi \in \mathbb{R} \) with periodic box constraints \(-\pi/2 \leq \phi \leq \pi/2\) (cf. Sec. 5.1), the thickness is represented as an integer multiple of a predefined thickness for each considered ply material. The ply material is a discrete value from a catalog of predefined materials.

9.1.2 Genetic variation operators

Genetic variation operators are applied to operate on four different levels of the genetic representation: On the parameterized shape, on the curves partitioning the structure into laminate sections, on the laminate, and on the level of single ply properties.

The following operators work on the level of shape parameters: Gaussian shape mutation varies the shape parameters in a candidate solution by adding a random number from a normal distribution \( \mathcal{N}(0, \sigma_{\text{shape}}) \). Arithmetic shape crossover performs an arithmetic crossover (see Sec. 4.3.2 in [143]) of the shape parameters \( \mathbf{s} x \) between two parent solutions.

On the level of curves, the operators are borrowed from Sec. 7.1.4: Add curve mutation adds an additional random curve to the arrangement. Remove curve mutation removes a curve from the arrangement. Gaussian curve muta-
Figure 9.1: Hierarchical representation of laminate sections on a structure of variable shape
modifies the position of curve-points by a Gaussian mutation. Geometric crossover is a generalization of the geometric face-property crossover: It splits the arrangement by a randomized, straight line, and recombines curves and face properties of the so generated segments. Curve split-and-splice crossover operates on the curves in the arrangement (cf. Chap. 3).

The laminate is varied by operators adapted from Sec. 6.1.5: Add ply mutation adds a random ply to the laminate stack. The position of insertion is determined by section rankings with gradient information. Remove ply mutation removes a ply from the laminate stack. Again, section rankings are used to determine the ply. Grow-ply mutation increases the region covered by a ply by adding it to additional sections. Shrink-ply mutation shrinks the region covered by a ply by removing it from sections. Move ply mutation moves a ply by adding an additional section and removing a section from the region it covers. Stacking mutation changes the laminate stacking by reordering the plies in a candidate solution. Laminate crossover exchanges the laminates between sections based on their locality in the arrangement. Thus it is a generalization of a the face property operator presented in Sec. 7.1.4.

On the level of ply properties, the following operators adapted from Sec. 6.1.5 apply: Gaussian ply angle mutation varies the ply angle of all plies by a Gaussian mutation. Ply material mutation exchanges the material of a random ply by a random material from a catalog. Ply thickness mutation replaces the thickness multiplier of a single ply by a random integer in the allowed range. The ply is determined by section rankings. SR1-ply-angle mutation optimizes the ply angles of all plies by a local deterministic search. Ply-property crossover exchanges ply materials between plies and makes an arithmetic crossover of ply thicknesses and ply angles between two parent solutions.

Following the conclusions of Chap. 8, the geometric modeling presented in Chap. 7 is simplified by restricting arrangement curves to start and end on the boundary of the domain Ω. Curves with arbitrary start points have a high probability to induce non-phenotypic edges. If a curve’s start or end point is located inside a face, this face cannot be split by the curve. Thus, a so called free edge is induced inside this face (Fig. 9.2). Obviously, free edges are never phenotypic since they do not separate faces. However, they participate in variation and are thus a potential source for non-robust geometric operations. The restriction to bound end points affects the set of legal variations in two ways: in variation operations on closed curves and in variation operations on curve end points.

Closed curves (loops, bubbles) are evidently no longer legal. Thus, variation operations inserting or operating on them as proposed in Chap. 7 are
Figure 9.2: Free edges induced by curves with arbitrary endpoints: Three curves induce the edges $e_1, \ldots, e_9$ and faces $f_1, f_2, f_3$. Two curves have endpoints inside a face and thus induce free edges $e_4, e_6, e_7$ and $e_9$.

disabled. This approach seems bearable, since closed curves are particularly helpful in applications with a single, isotropic material. With anisotropic materials a void region inside a homogeneous face would inevitably disrupt a load path. Notably, disabling closed curves does not prohibit the creation of void regions (Fig. 9.3). The creation of void regions by bound curves allows for distinct face properties surrounding the void at the cost of increased parametric requirements.

Variation has to detect and consider bound end points. Legal variations on them are restricted to shifting them on the boundary but not into the domain. This is achieved by accepting only variation components tangential to the boundary. Special attention is required for points in a corner of the domain. There, two complementary variation directions are legal.

9.2 Numerical Experiments

9.2.1 Single-ply vibrating plate

In order to investigate the effect of an arrangement-based partition of the geometry with a predefined setup of laminate sections, the vibrating plate experiment of Sec. 6.3.1 is repeated. For comparison, an optimization of the shape (normal to the plane) is not considered. In the here presented application, the geometry shall be partitioned by an arrangement of maximally three
Figure 9.3: Representation of void includes (bubbles) by a single closed curve compared to two bound curves

spline curves with three points each. Each curve has four movable coordinates (one for each endpoint, two for the midpoints). Thus, at most twelve genes are required to encode the geometry. These curves shall be used to define laminate sections on the structure. Only a single predefined laminate section remains, i.e. the one covered by the additional mass layer. Materials, geometry, modeling, and the fitness formulation (6.14) are the same as in Sec. 6.3.1.

Ten independent optimization runs are carried out. For each run, the pool size is set to 250 individuals, four demes are employed, and migration frequency is set to 750 evaluations (the setup corresponds to the one of Sec. 6.3.1). The application rates for the operators are set to 80% crossover (20% for each of split-and-splice curve, geometric, laminate, and ply property crossover) and 20% mutation (5% for move ply, 5% for Gaussian-curve, 2.5% for ply angle, 2.5% for SR1-ply-angle, and 1.25% for each of add curve, remove curve, grow-ply, and shrink ply mutation). Operators covering aspects not relevant for the application are disabled (variations of the number of plies, stacking sequence, or shape attributes).

Results and Discussion: Convergence plots for the ten runs are depicted in Fig. 9.4. Each run finds a feasible solution with a mass of 0.436kg. The first natural frequencies of the best solutions are in a range from 2.482 to 2.803Hz. The best solution for each run is illustrated in Fig. 9.5. The solutions in Fig. 9.5(d) and Fig. 9.5(e), respectively Fig. 9.5(f) and Fig. 9.5(h) are identical. Six out of ten runs do not reinforce the mass section, five feature a hole in the reinforcement ply. The ply angles are adapted to the shape of the reinforcement.
9.2 Numerical Experiments

The optimization method can profit from the increased freedom in design: The solutions are all fitter than those found on a partitioned structure with 50 zones in Sec. 6.3.1. However, the frequencies of the best solutions presented here show a considerably higher spread (standard deviation 0.107 Hz) than those found on predefined sections (standard deviation 0.034 Hz). A slow down in the convergence can be observed: Whilst the best solutions on predefined sections are found after at most 7000 evaluations, some of the runs with movable section boundaries do not show stagnation within the 15000 evaluations investigated here.

9.2.2 Shape adaptive vibrating plate

The vibrating plate experiment from Sec. 6.3.2 is extended by additional shape parameters. Therefore, the geometry is defined by a spline loft characterized by a point $P$ (Fig. 9.6). The Cartesian coordinates of this point, i.e. $P_x$, $P_y$, and $P_z$ serve as shape variables. The support including the outer geometry, the materials, as well as the mass stay the same as in Sec. 6.3.2 (cf. Fig. 6.4). The objective is the same is in the fixed shape application (6.14). The scaling and constraint values are set to $f_s = 150 Hz$, $M_{max} = 0.48275 kg$, $p = 10$, and $M_s = 0.05 kg$.

The optimization is carried out for 100000 evaluations with a pool size of 300 individuals in 5 demes. The operator rates are set to 16.1% for each of the crossover operators and 1.5% for each mutation operator except for material mutation which is disabled (0%).
Figure 9.5: Best individuals identified in 10 different runs for the single-ply vibrating plate experiment with their first natural frequency $f_0$, mass $M$, and ply-angle $\phi$. 
Figure 9.6: Shape adaptive vibrating plate experiment

(a) Fitness  
(b) First natural frequency  
(c) Mass  
(d) Number of plies

Figure 9.7: Convergence plots for the shape adaptive vibrating plate experiment
Figure 9.8: Shape of the best solution for the shape adaptive vibrating plate experiment.

(a) Patch 1: thickness multiplier $t = 1$, ply angle $\phi = 73.25^\circ$

(b) Patch 2: thickness multiplier $t = 3$, ply angle $\phi = 88.74^\circ$

Figure 9.9: Patches of the best solution for the shape adaptive vibrating plate experiment.
9.2 Numerical Experiments

Results and discussion: Convergence plots are depicted in Fig. 9.7. The best solution is found after 47766 evaluations as offspring of $SR1$-ply-angle mutation. It features two plies (Fig. 9.9). Its natural frequency is $f_0 = 140.082\,\text{Hz}$, the mass $M = 0.482604$. The shape of the best solution (Fig. 9.8) shows an excentric elevation in the right half of the structure (i.e. the one without the additional mass).

A curved shape leads to a considerable increase in the plates bending stiffness. This can be seen in the first mode (Fig. 9.10) of the best solution.: Although there is only one ply covering the region with the highest curvature, it does almost not participate in the first mode. The curved shape leads to an increase of the first natural frequency by a factor of about six when compared to the results obtained in Sec. 6.3.2. Nonetheless, each curved geometry has obviously a higher mass than the flat plate. Hence, the optimization has to balance the increased stiffness against a violation of the mass constraint. This could explain the relatively low number of plies. The reinforcement in the region of the mass, as well as its angle and thickness seems justified. However, its asymmetric shape is not easily interpretable and may be affected by the asymmetric shape of the structure itself.

9.2.3 Cantilever

A cantilever structure with a given mass shall be optimized for minimal compliance. The structure is illustrated in Fig. 9.11. The shape is defined as a free-form surface parameterized by a set of nine points. The structure is fixed at the edge $F'EF$ ($u_x = u_y = u_z = 0$). The edge $B'AB$ is loaded by
a distributed line load of 10N/m in the negative z-direction. The shape is symmetric with respect to the x-z-plane. Hence, a half-model spanned by the points A, B, C, D, E, and F is used with symmetry boundary conditions on the edge ACE. The shape is variable, i.e. the z-coordinates of the points C, D, E, and F are decision variables. The arrangement is realized in the x-y-plane and projected onto the structure. The design domain is defined by a rectangle with length L = 1m and width H/2 = 0.5m. Laminate sections are induced by an arrangement of at most 20 curves with each at most 6 points. Thus, at most 200 genes are required to parameterize the shape of the reinforcements.

The structure is initially covered by a layer of a very compliant fill material (isotropic, Young’s modulus $E = 1GPa$, Poisson’s ratio $\nu = 0.3$, thickness 0.015mm, density $10^{-18} kg/m^3$). Reinforcements shall be realized by at most 12 layers of fiber reinforced materials, i.e. material 1 (Tab. B.1(a), thickness 0.15mm), material 2 (Tab. B.1(b), thickness 0.15mm), and material 3 (Tab. B.1(c), thickness 0.2mm). Each ply may have at most three times the thickness of the raw material.

The objective is to minimize the work of the external forces $W_e$ subject to a mass constraint. A scalar fitness is assembled by the exterior penalty approach (Sec. 2.4.2):

$$\min_{x} F(x) = W_e(x) + p \left[ \max \left\{ 0, \frac{M(x) - M_{\text{max}}}{M_{\text{s}}} \right\} \right]^2$$  \hspace{1cm} (9.1)

The mass is restrained to $M_{\text{max}} = 0.5kg$ ($M_{\text{s}} = 0.05kg$), the penalty factor is set to $p = 10$.

The structural model is meshed with an unstructured mesh by an advancing front algorithm. A structured mesh turned out to produce higher element distortions especially on shapes with small local curvature. An unstructured mesh thus allows for an unbiased shape optimization. Notably, the number of elements is not constant but depends on the actual shape. Since not each face in the geometric arrangement is meshed separately, very small faces may not be mapped to element properties in the simulation model.

The optimization is carried out with a pool size set to 400 individuals in 6 demes. The operator rates are set to 80% crossover and 20% mutation (equally distributed onto the corresponding variation operators). The optimization is stopped after 98344 evaluations.

**Results and discussion:** Convergence plots are depicted in Fig. 9.12. The best solution found has a feasible mass of 0.497kg. The external work evaluates to $4.16385 \cdot 10^{-5} J$. It features 12 plies whereof one is smaller than
the element size and thus not present in the simulation model. The shape and reinforcements are illustrated in Fig. 9.13. The reinforcements consist of high-modulus material 2 and woven material 3. The unidirectional high-strength material 1 is not employed. A single reinforcement covers the complete structure (Patch 8, Fig. 9.13(h)) whilst the other reinforcements cover the region close to the load introduction.

The shape of the best solution shows a curved cross section ($CD$ and $ EF$). This increases the moment of inertia and thus stiffens the structure against bending moments in the $y$-direction. The shape parameterization enforces a straight edge at the load introduction (section $AB$). This region is thus compliant in bending in the $x$ but also the $y$ direction. A large portion of reinforcements is applied in this almost planar tip region close to the load introduction. The unidirectional patches 1, 2, 7, 9, and 10 feature ply angles which reinforce the structure against bending moments in the $x$-direction. The patches 3, 6, and 11 carry tension forces up to the region of cross section $CD$ where the shape is stiffened by the curvature.

The optimization is possibly not fully converged. Hence, the mass constraint is not yet fully exploited. The possibility to leave regions empty in order to employ reinforcements in truss-like structures is not utilized by the optimization.
Figure 9.12: Convergence plots for the shape adaptive cantilever experiment
(a) Patch 1: $\phi = -85.9^\circ, t = 3, m = 2$
(b) Patch 2: $\phi = 85.9^\circ, t = 3, m = 2$
(c) Patch 3: $\phi = 6.4^\circ, t = 3, m = 3$
(d) Patch 4: $\phi = 43.1^\circ, t = 2, m = 3$
(e) Patch 5: $\phi = -52.3^\circ, t = 2, m = 2$
(f) Patch 6: $\phi = -11.9^\circ, t = 2, m = 3$
(g) Patch 7: $\phi = -88.2^\circ, t = 3, m = 2$
(h) Patch 8: $\phi = 7.2^\circ, t = 2, m = 2$
(i) Patch 9: $\phi = -85.3^\circ, t = 3, m = 2$
(j) Patch 10: $\phi = 84.6^\circ, t = 3, m = 2$
(k) Patch 11: $\phi = 11.7^\circ, t = 2, m = 2$

Figure 9.13: Patches with ply angle $\phi$, material $m$ and thickness multiplier $t$ on the shape of the best solution found for the shape adaptive cantilever problem
9.3 Conclusions

A hierarchical representation aimed at the combined optimization of laminate and shape attributes of a structure is presented. By enabling void as a legal face property, the method allows for topological changes as well.

Experiments on the single-ply vibrating plate problem indicate the method’s ability to exploit the increased freedom in design: the arrangement-based laminate optimization is able to find fitter solutions than a laminate optimization on a fixed geometric partition. On more complex problems, the optimization steadily evolves improvements from initial designs. It is able to keep track of cross-influences between shape and laminate variation. This is illustrated in the solutions found by the optimization. They show laminate and geometric properties which complement themselves, i.e. the laminate is mainly reinforced in regions where the variable shape exposes weak properties and vice versa. However, the promising results of the single-ply vibrating plate experiment cannot be generalized to more complex structures. The method fails at finding optimal designs in terms of truss-like, or fully-stressed designs. This, potentially because truss members are difficult to model by the investigated boundary-based representation. They have to be defined by two boundary curves, one on each side. Once established, non-disruptive shape changes on thin structural members require a concurrent change of these boundary curves without eliminating the area in-between. Thus, genetic variation has to make a lucky guess of correlation values for a potentially large number of genes encoding these boundaries.

The method couples laminate sections and arrangement geometry. This allows to generate locally varying laminates by a limited set of geometric parameters when compared to encoding the shape of each ply separately. Nonetheless, variation on single reinforcements is now restricted to already existing geometric faces. These faces are not homogeneously sized and distributed inside the domain. Hence, variation operating on the laminate level is influenced by the genetic information on the arrangement level. Thus, a successful adaption of the laminate to local demands requires a lucky sequence of elementary variation operations on both, the arrangement and laminate level.
Chapter 10

Conclusions and Outlook

This chapter summarizes the results and findings. It deduces final conclusions and gives recommendations for further research.

10.1 Concluding remarks

The automated optimization of fiber reinforced composite constructions is addressed in three different disciplines: the optimization of directions of anisotropic reinforcements (ply angles), the optimization of locally varying laminate properties (materials, thicknesses, orientations), and finally the optimization of laminates on structures with variable shape and topology.

All investigations focus on methods derived from a generic EA scheme. An adaption of this algorithm to the particular traits of the optimization of laminated composites is done in two different domains: In the genetic representation and in the incorporation of deterministic local searches or gradient information.

The use of variable-length representations is investigated in the context of laminates with locally varying thicknesses. There, an EA is able to evolve the number of reinforcements as well as their attributes. Special attention is payed to variation operators: a variable-length split-and-splice crossover is created incorporating constraints on genotype length. It avoids a source of illegal solutions in already existing approaches.

Gradient information is incorporated to reduce the number of evaluations and increase the solution quality. In order to embed gradient-based, local search methods in an EA scheme, the existing parallelism in the algorithm is extended to the variation state. This is achieved by avoiding synchroniza-
tion points in the algorithm design. A niching strategy helps to keep book of already discovered local optimal solutions. The stochastic component is then focused on unexplored regions of the design space. This asynchronous, parallel, memetic algorithm turns out to outperform conventional EAs in test functions with a moderate number of local optima.

The asynchronous algorithm is applied to ply angle optimization problems. These problems offer the evaluation of sensitivity information at relatively low additional cost. The parameterization of global plies allows for the optimization of structures with locally varying laminates without disturbing the cohesion. The method is able to reproducibly find global optimal designs in simple academic benchmark problems. The method is applicable to common engineering problems which is demonstrated in two case studies.

A structured, variable-length genotype for the optimization of laminates on geometrically partitioned shells is presented. It borrows from the patch concept and graph-based variation operations of prior approaches. The incorporation of gradient information in the variation operations significantly reduces the number of evaluations required and improves the reproducibility of the results. A case study demonstrates the methods ability to find feasible designs in typical engineering problems.

An arrangement-based genotype for TOD is investigated. It evolves geometric regions bound by spline curves and assigns void or material thereon. The method makes use of graph-concepts to establish a connected load path. It is able to evolve feasible designs. Nonetheless, it is not competitive with existing approaches. Investigations identify the non-robustness of geometric operations as one reason for the inefficiency of the approach. These are avoided by shape-healing mechanisms and a reduction to bound spline curves in further developments.

A hierarchical representation aimed at the combined shape, laminate, and topology optimization is presented. The method projects an arrangement-based partition of laminate sections onto a parameterized shape. In simple academic examples on a fixed shape the arrangement-based laminate optimization method is able to evolve fitter designs than the before introduced approach with predefined geometric partitions. Nonetheless, the arrangement-based approach requires more evaluations and repeated runs on the same problem show a considerably widened spread. Experiments on variable shape structures show enormous improvements of the mechanical properties when compared to fixed shape optimizations. This is due to the much higher impact of shape variations on the structures performance when compared with laminate variations. The optimization results indicate a problem specific complementary tailoring of local laminate and shape
attributes. However, the method is not able to find optimal designs in terms of truss-like structures.

The investigation of representation schemes in evolutionary structural optimization lead from purely parametric applications in conventional real-encoded GAs, over heterogeneous parameter lists [143], to graph-based applications [89, 218], and tree-representations in CAD-environments [257]. The here presented work is motivated by results and research needs presented by [89, 262]. The increasing complexity and abstraction in genotype representations discovers difficulties in the creation of affordable and non-disruptive variation operations. The tailoring of a linear representation to the needs of ply angle optimization in Chap. 6 provides promising results with a very limited set of only three variation operations of affordable complexity. The structured representation of locally varying laminates in Chap. 6 is based on already twelve variation operators. However, the abandoning of a purely graph-based genotype of [89] helps to significantly reduce the complexity of each single variation operation. The arrangement-based representation of a variable topology (Chap. 7) involves potentially non-robust geometric operations and is thus dominated by expensive computations. The influence of these on runtime can be damped by an asynchronous scheme which runs not only evaluation but also variation in parallel. This is part of the motivation for the algorithm proposed in Chap. 4. Although, the arrangement representation is simplified for a combined shape and laminate optimization (Chap. 9) the tight coupling between different hierarchical levels affects the chance for successful variations. This, because such variation often requires a concurrent change on multiple levels or multiple attributes of the structure (as detailed in Sec. 9.3).

The presented work illustrates the high potential of combined optimization approaches in evolutionary optimization of laminated composite structures. The seek for creative, autonomous problem solvers along a path of increasing abstraction in the genotype representation at some point leaves the affordable region what concerns computational requirements. Thus, freedom in automated design – which goes hand in hand with abstraction in the representation and creativity in the solutions –, and available resources have to be balanced in order to arrive at a method which is affordable for a given environment. For practical applications, the two methods presented in Chap. 5 and Chap. 6 provide each a particular advantage over existing approaches: the ply angle optimization method presented in Chap. 5 employs a parameterization of an existing design where only reinforcement orientations are optimized. Although the same problem can be addressed by optimization tools available in
commercial and non-commercial software, the presented method improves the state-of-the-art in the parameterization of global plies which avoids a cumbersome post-processing, in its robust global search capabilities, and in its acceptable efficiency provided by embedded local searches. The laminate optimization method presented in Chap. 6 considerably increases the freedom in design over existing approaches. The variable-length representation can be understood as a self-parameterizing design optimization. Thus, it is able to evolve locally varying laminates which are not part of an initial design. The ply-wise representation is easily translated to a lamination plan. The improvements induced by asynchronism, by a more compact genotype, and by embedding gradient information in the variation state considerably reduce the computational requirements of previous methods.

For TOD a method operating on boundary-based, geometric patch shapes leads to high parametric requirements. Thus, to exploit the link between laminate and topology optimization a limitation to primitive geometric members in the encoding of the shape attribute of each reinforcement could significantly improve the seek for superior designs.

10.2 Outlook

Apart from the already outlined detailed needs and possibilities for further research in the corresponding chapters, this section shall illustrate three ideas emerging from this thesis project.

10.2.1 Similarity evaluation by overlays

Overlays, as presented in Sec. 7.1.4 offer interesting possibilities to operate on an abstract level on geometric constructions. In order to record, control, or increase diversity in an optimization, distance or similarity metrics have to employed (cf. e.g. Sec. 4.2.3). Overlays offer the possibility to evaluate similarities independent from the representation scheme encoding the structure (Fig. 10.1): The ratio of the common volume of two structures and their total volume, i.e. their intersection divided by their union, is a natural way to determine similarities. This ratio gives an easily interpretable number in the interval \([0, 1]\). Identity evaluates to one, whilst structures sharing nothing will have a similarity value of zero. A study on an EA employing similarities to control population diversity is presented in Appendix A. However, the direct application of a similarity metric based on overlays in this context requires further research. The overlay-based evaluation of each similarity pairing in a
population is computationally expensive due to the high number of pairings \((n^2/2 - n/2)\) for \(n\) being the population size).

Further research is necessary to identify and exploit potential use of similarities and to investigate possibilities to reduce the number of similarity computations in the presented applications in order to make them appropriate for computationally more expensive overlays.

### 10.2.2 Manufacturing considerations

The choice of a material system is always related to a choice of a manufacturing process. Manufacturing related aspects offer at least two possible extensions of the here investigated methods:

**Extension to other manufacturing processes** : This thesis focuses on structural components manufacturable by fabric or prepreg lay-ups. Other manufacturing methods like tow placement, tape laying, or filament winding may allow for completely different designs. An optimization method based on the genetic encoding of primitive features of these designs could borrow concepts developed in this thesis.

**Embedding of manufacturing related aspects** : Particularly if shape changes are involved, the incorporation of manufacturing related aspects in the structural model gains in importance. Such aspects could for instance include the simulation of draping processes to determine locally changing ply angles and thicknesses, the prediction of wrinkling problems of the reinforcement fabrics, or the simulation of resin injection processes. Thus, they are not directly related to the optimization method but more to the features of the simulations employed in the structural model. A requirement – and at the
same time one of the main obstacles – for the consideration of manufacturing aspects is the possibility to accurately, efficiently, and robustly model the relevant structural properties in a fully automated simulation environment.

10.2.3 The surface concept

The concept of face-based patch representations on geometrically partitioned shells (Chap. 6) can be fused with ideas presented by Iuspa and Ruocco [126]: A generative genetic representation for topology optimization on stringer reinforced composite panels can be developed. There, the number and position of stringers would be fixed in a ground structure (Fig. 10.2). An optimization of variable-thickness laminate properties on the ground structure then determines which stringers become physically existent and which ones may vanish.

The surface concept may be used to extend the existing laminate optimization method: A surface is a group of adjacent faces associated with a reference normal and draping direction. Hence, the most simple geometric face in a shell structure has two surfaces, i.e. a top and a bottom surface each covering the same geometry but distinct in their normal direction. The set of surfaces associated with a ground structure of a stringer reinforced panel like the one drawn in Fig. 10.2 are determined by geometric properties. Faces may be grouped to surfaces based on their angle in-between: Whilst a draping process may allow for plies covering 90° corners, a 180° bend in a ply is infeasible. Patches are then applied to surfaces. The sequence of patches already applied determines the possibilities for further variation. A representation based on the surface concept allows to consider the cohesion of the structure by global plies and at the same time avoids illegal crossings of plies in the cross section (Fig. 10.3).

This method requires the realization of three components in the same framework: An automated method to detect surfaces from geometric properties of a ground structure and already applied patches, a sophisticated handling of ply informations and their mapping to surfaces, and the consideration of draping effects in order to determine properties of bent plies.
Figure 10.3: Representation of a stringer reinforced panel topology with six patches. One stringer of the ground structure is not covered and hence inexistent.
Appendix A

Similarity-based Adaptive Fitness Sharing

The here presented and investigated algorithms lead to the fitness sharing concept employed in Chap. 4.

Population diversity is a crucial value in the analysis, design, and application of Evolutionary Algorithms. Different measures on population diversity are used to analyze or guide the evolutionary search: a theoretical analysis on the evolution of population diversity measures is presented e.g. by Leung et al. [156]. Lacevic et al. [150] shows weaknesses of common used diversity metrics and proposes new ones, based on a singular value decomposition of a distance matrix containing the distances between all population members. Applications of diversity metrics to improve the convergence behavior are investigated by [211, 188]. A more sophisticated consideration of diversity by the use of clustering information is proposed by [157, 236]. Applications of diversity-based algorithms are popular in the field of multi-objective optimization in order to attain a good approximation of the Pareto-optimal set (e.g. [268, 239]). Finally, custom-tailored diversity concepts can be found in structural optimization application (e.g. [20]).

Barker and Martin [25] show that under certain assumptions the expectation value of a distance based population diversity metric converges to a fixed point during the evolutionary search. If no mutation occurs, this fixed point is equal to zero where all individuals in the population become identical. The preliminary assumptions and therefore the existence as well as the exact value of the fixed point only depend on the input parameters in the mutation operator for their setup. Although missing this theoretical basis,
early considerations on maintaining population diversity in order to prevent premature convergence in a local optimum have been discussed: the concept of fitness sharing, first proposed by [120] and later elaborated by [96], is intentionally designed to identify multiple local optima in non-convex, multi-modal objective landscapes by a single optimization run. Since its first occurrence in literature, the concept has been developed and cited under the more general term *niching method* together with other approaches to comprise the population density or to control the population diversity (see [217] and references therein).

Fitness sharing is inspired by natural evolution, where the resources are limited and therefore reduce the payoff for species living in densely populated niches. In such niches fitness sharing introduces selection pressure towards more diversity by lowering each individuals fitness. Typically, the shared fitness for maximization problems is

\[ F'_i = \frac{F_i}{m_i}, \tag{A.1} \]

where the *niche count* \( m_i \) is an approximate number of individuals living in the same niche like individual \( i \) whose fitness \( F_i \) is shared. According to [96] the niche count is computed based on a distance metric \( d_{ij} \) (distance between individual \( i \) and \( j \)) in the search space transformed by a *sharing function* \( \text{sh}(d_{ij}) \) for all \( n \) individuals in a population:

\[ m_i = \sum_{j} \text{sh}(d_{ij}) \tag{A.2} \]

The common used sharing function is parameterized by a problem dependent niche radius \( \sigma_s \): Individuals outside a sphere \( S \) with radius \( \sigma_s \) around individual \( i \) in genotype space do not contribute to its niche count. From the distance \( \sigma_s \) towards the center of \( S \) the contribution of the corresponding individual increases from zero (on the boundary of \( S \)) to one in the center. Hence, the sharing function expresses a similarity with value one for identity and zero for sufficiently dissimilar.

\[ \text{sh}(d_{ij}) = \begin{cases} 1 - \left( \frac{d_{ij}}{\sigma_s} \right)^\alpha & \text{if } d_{ij} \leq \sigma_s \\ 0 & \text{otherwise} \end{cases} \tag{A.3} \]

\( \alpha \) is a real valued tuning parameter for a problem dependent adaption of the power law (cf. Fig. 4 in [96]). The very basic concept of fitness sharing assumes a number of subpopulations covering the peaks in the fitness landscape, but does not provide any information on the number or position of these peaks.
Since there is usually no prior knowledge on the number, location, and size of the niches, a major problem in the application of fitness sharing is the determination of the problem dependent niche radius $\sigma$. Various efforts have been carried out in the recent years, to identify the location of niches or to automatically adapt the niche radius during the optimization run: [67] proposes a heuristic method to estimate an optimal sharing radius on the number of niches for both, genotypic and phenotypic distance metrics in binary encoded Genetic Algorithms. Niching methods have been investigated in the field of Evolution Strategies, like the well known CMA-ES, where – based on the estimated covariance matrix – the size and shape of niches may be locally adapted to the fitness landscape ([228, 229]). In the field of Genetic Algorithms, investigations on estimating shapes or niche specific radii has been investigated by [85, 86, 87].

An estimation on the location of niches may be found e.g. in [71]. Asymmetric sharing as proposed by [94] considers the niche radius as an additional decision variable and hence incorporates it as a gene in the genotype representation. [68] propose a characterization method providing an estimation on the mean and the standard deviation of the number of niches as a function of niche radius and population size. In [69] they include their findings into a method called *dynamic niching* designed to dynamically identify the number and position of niches. Although not explicitly mentioned, the *Dynamic Species Identification Algorithm* (Alg. 1 in [69]) is an implementation of the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm originally proposed by [77] and meanwhile quite common as a partitioning clustering algorithm in data analysis. This study investigates a *dynamic niching* algorithm extended by the following concepts:

- **Fitness formulation**: The original fitness sharing concepts including the dynamic niching algorithm focus on unconstrained maximization problems. Within this study a generalization to a non-scalar fitness concept aimed at constrained and unconstrained, single-objective minimization or maximization problems is proposed.

- **Similarity metric**: Although not explicitly named, the concept of a sharing function introduces a normed similarity metric. Thus, similarity is an intrinsic component of any fitness sharing method and is usually computed from distance information. It can be seen, that for certain problems in structural optimization similarity information is more intuitive to find than a distance metric. Thus, the algorithm is redrafted to employ a similarity metric instead of a distance metric in the sharing states.
• **Adaptive niche-radius control:** A heuristic feed-back control rule for the dynamic adaption of the niche-radius is proposed and investigated on a test-bed of well-known constrained and unconstrained benchmark functions.

### A.1 Method

#### A.1.1 Fitness Sharing Evolutionary Algorithm

**Algorithm 28** Evolutionary Algorithm Scheme

1. Initialize a population $M(0) \leftarrow \{(x_i, F(x_i))\}$
2. Initialize the iteration counter $t \leftarrow 0$
3. Compute $F$ for each $x_i$ in $M(0)$
4. while continue($M(t), t$) = 1 do
   5. Compute niche count $m(x, M(t))$ for each $x_i$ in $M(t)$
   6. $M' \leftarrow$ mating selection from $M(t)$
   7. $M'' \leftarrow$ mutation and recombination from $M'$
   8. Compute $F$ for each $x_i$ in $M''$
   9. $m(x, M(t)) \leftarrow 1$ for each $x_i$ in $M''$
10. $M(t+1) \leftarrow$ environmental selection from $M(t)$ and $M''$
11. Increment $t \leftarrow t + 1$
12. end while
13. Find the best solution $x^*$ in $M(t)$

This study investigates an EA in the form of a general Genetic Algorithm as outlined in Alg. 28 but with additional sharing states (5 and 9). In the context of Evolutionary Optimization, the information $x$ is referred to as **genotype** and $X$ as genotype space respectively. The states 1 and 7 in Alg. 28 depend on the structure of the genotype $x$ and are therefore not discussed in this context. Candidate solutions are mapped from genotype space $X$ to a phenotype space $Y$. A fitness is assigned according to the properties evident in $Y$.

SUS is implemented as mating selection. Environmental selection is done by comma selection: The offspring $M''$ replace the parents $M(t)$. A niche-elitist strategy is employed, i.e. the best individual of each identified niche is not replaced. The algorithm is stopped if $t$ is larger than a predefined number or there is no improvement in $t$ iterations.

A fitness $F$ is introduced based on the separation approach presented in Sec. 2.4.1. Objective and constraint violation are shared based on the original
A.1 Method

idea of [96]. The concept is extended to minimization problems and positive as well as negative objective values are considered in the computation of the shared objective $F_s(x)$ and constraint violation $c_s(x)$:

\[ S(F) = \begin{cases} 
1 & \text{if } F(x) \text{ has to be minimized} \\
-1 & \text{if } F(x) \text{ has to be maximized} 
\end{cases} \quad (A.4) \]

\[ F_s(x) = F(x) \cdot m(x, M(t))^{S(F) \cdot \text{sgn}(F(x))} \quad (A.5) \]

\[ c_s(x) = c(x) \cdot m(x, M(t)) \quad (A.6) \]

The niche count $m(x, M(t))$ is computed from the genotype $x$ whose fitness is shared and the population $M(t)$ in generation $t$ (cf. Sec. A.1.2). Fitness sharing is only performed in the mating selection state (Alg. 28, state 6). In order to compare individuals from parent and offspring generation, the niche count is set to 1 before environmental selection (Alg. 28, state 10).

The following rules apply to find an order in the two candidate solutions $x_1$ and $x_2$ for $x_1, x_2 \in X$:

\[ F(1x) < F(2x) \iff \begin{cases} 
S(F)F_s(1x) < S(F)F_s(2x) & \text{if } c_s(1x) = c_s(2x) \\
c_s(1x) < c_s(2x) & \text{otherwise} 
\end{cases} \quad (A.7) \]

\[ F(1x) = F(2x) \iff (c_s(1x) = c_s(2x)) \land (F_s(1x) = F_s(2x)) \quad (A.8) \]

The remaining comparison operations can easily be derived from equations A.7 and A.8. The resulting fitness $F$ has to be minimized, i.e. an ascending order in $F$ equals the order from the best to the worst solution in a population. $F$ strictly prefers feasible solutions over non-feasible ones.

A.1.2 The Similarity Concept

The here introduced concept of similarity is a natural component in the motivation for the original sharing concepts. But, contrary to the usually employed distance metrics a similarity metric $s_{ij} = s(x_i, x_j)$ is assumed to exist with the following properties:

\[ s_{ij} = \begin{cases} 
1 & \text{if } x_i = x_j \\
0 & \text{if no resemblance in } x_i, x_j \\
0 < s_{ij} < 1 & \text{otherwise} 
\end{cases} \quad (A.9) \]

The metric $s_{ij}$ may either exist in the genotype space $X$ – comparable to the distance metric in common fitness sharing schemes – or in the phenotype space $Y$. Two individuals $x_i$ and $x_j$ are in the same niche if their similarity
$s_{ij} \geq \sigma_s$. Thus, the niche radius $\sigma_s$ is no longer expressed as an upper bound for a distance in $\mathbb{X}$ but as a lower bound in the normed similarity interval $[0, 1]$.

**Algorithm 29** Dynamic Niche Identification

**Require:** Population $M$

**Require:** Niche radius $\sigma_s$

**Ensure:** List of niches $N$

1: Initialize niches $N \leftarrow \{}$
2: Initialize marked individuals $Q \leftarrow \{}$
3: Sort population by fitness $M_s \leftarrow \text{sort}(M)$
4: for $x_i \in M_s$ do
5: \quad if $x_i \notin Q$ then
6: \quad \quad Initialize new niche $N_i \leftarrow x_i$
7: \quad \quad for $x_j \in \{x \in M_s \mid x \notin Q\}$ do
8: \quad \quad \quad if $s(x_i, x_j) \geq \sigma_s$ then
9: \quad \quad \quad \quad Add $x_j$ to current niche $N_i \leftarrow N_i \cap \{x_j\}$
10: \quad \quad \quad Mark $x_j$, $Q \leftarrow Q \cap \{x_j\}$
11: \quad \quad end if
12: \quad end for
13: \quad if $|N_i| > 1$ then
14: \quad \quad Add current niche to niches $N \leftarrow N \cap \{N_i\}$
15: \quad end if
16: end if
17: end for

**A.1.3 Diversity Reward and Heuristic Feedback-Control of Niche Radius**

If niching is employed to counteract genetic drift and therewith premature convergence, the choice of the niche radius is crucial. Convergence (including premature convergence) leads to an assimilation of individuals in the population, thus the similarity of individuals increases over generations and therewith the number of individuals in the same niche increases. Fitness sharing is not designed nor able to eliminate the selection pressure towards fitter solutions. Thus, assuming there are niches occupied by fit individuals, these niches are attractive for further breeding and the number of individuals inside will increase over time. A theoretical analysis on this observation is already given in [96]: There, the expected number of individuals in each niche is proportional
to the height of the corresponding peak in the fitness landscape. Because the fitness sharing penalty is the same for all individuals inside a given niche and their raw fitness is approximately in the same range genetic drift is observable inside of each niche as an assimilation of all its individuals. Thus, from the above explanations and observations, there is a strong tendency towards a few niches partitioning the population and consisting of identical individuals after sufficiently many generations. Due to the loss of genetic information inside of each niche the potential for further improvement rapidly decreases. If the niches are approximately equally populated, fitness sharing is no longer able to counteract this effect because the penalty induced is the same for all individuals.

In the following a combination of two adaption strategies counteracting the described reasons for a loss of genetic diversity are described. They found on the available similarity information computed by the dynamic niche identification algorithm (Alg. 29): The similarity of a population can be represented as a square, symmetric matrix $S = (s_{ij})$. From the definition of $s_{ij}$ (eq. A.9), the diagonal entries of $S$ have to be equal to 1 and its off-diagonal entries are in $[0, 1]$. The dynamic niche identification algorithm computes slices of $S$. Only in a worst case (in terms of numerical effort, where the maximum similarity in the population is lower than $\sigma_s$, the full similarity matrix is computed. All off-diagonal values $s_{ij}$ computed during one application of Alg. 29 are further referred to as the visible part $S_v$ of $S$. The largest and smallest visible off-diagonal value $s_{min} = \min(S_v)$ and $s_{max} = \max(S_v)$ contain information on the diversity inside the population. To eliminate the risk to identify only a few large niches, the niche radius $\sigma_s$ is guided towards:

$$\sigma_t = \alpha s_{min} + (1 - \alpha) s_{max}, \quad \alpha \in [0, 1]$$  \hfill (A.10)

The niche radius $\sigma_{s}^{t+1}$ for the next generation is then computed as:

$$\sigma_{s}^{t+1} = \lambda \sigma_t + (1 - \lambda) \sigma_{s}^{t}, \quad \lambda \in [0, 1]$$  \hfill (A.11)

From our numerical experiments $\alpha = 0.25$ and $\lambda = 0.6$ turned out to be a suitable choice and are used for all the numerical examples in the following. For the assumption of a converging algorithm, where all individuals will concentrate at the same location in $\mathbb{X}$, thus, $s_{ij}$ and therewith $s_{min}$ and $s_{max}$ converge in 1. Therefore, typically an increasing niche radius is observed, although it is not necessarily monotonically increasing.

In order to increase the pressure on densely populated niches with low diversity, niches with a high diversity are rewarded. This is achieved by computing the standard deviation of the similarities inside each niche:

$$s_{N,k} = \text{std} \left\{ \{s_{ij}\} \right\}, \text{ for individual } i, j \text{ in niche } k$$  \hfill (A.12)
The niche count $m_k$ for all individuals in the niche $k$ (containing $r$ individuals) is then computed as

$$m_k = r^{-sN,k}.$$  \hspace{1cm} (A.13)

### A.2 Validation

The method is validated on a set of common used benchmark problems (Tab. A.1). F1 and F2 are taken from De Jong’s test function suite [66], F3 is the Michalewicz Function ([178]) and F4 is the Rastrigin’s Function ([182]). The constrained test functions C1 and C2 are taken from [80], C3 and C4 are from [119].

#### A.2.1 Method

The search space for all these problems is real-valued. A common Evolutionary Algorithm with One-Point-Crossover (at a relative application rate of 0.9) and a Gaussian-Mutation-Operator (rate 0.1) is used in the variation state. For $n$ being the number of dimensions in the search space, the population size is set to $4n$, and the number of generations to $8n$ (unconstrained), and $15n$ (constrained) respectively. The variance $\sigma$ in the Gaussian-Mutation is isotropic and set to $0.05 \cdot d_{max}$. $d_{max}$ is the maximum possible distance between two individuals in the genotype space, i.e. the length of the diagonal of the search space. Since the search space is not bound for all problems, $d_{max}$ is assumed as follows: F1: 63.25, F2: 89.44, F3: 9.93, F4: 45.79, C1: 17.61, C2: 15.49, C3: 52.92, and C4: 63.25.

The similarity function is defined as:

$$s_{ij} = e^{-\frac{\|x_i - x_j\|}{d_{max}}}$$ \hspace{1cm} (A.14)

Note that the minimal achievable similarity in the bounded search spaces is not zero, but $e^{-1}$ at $\|x_i - x_j\| = d_{max}$, thus conserving a robustness margin for the partially unbound problems C1 and C2.

Three different configurations are examined on these benchmark functions:

- **No Sharing (NS):** As a reference, an algorithm without sharing is used.

- **Fitness Sharing (FS):** This configuration consists of the dynamic niching algorithm without adaption of the niche radius.
- **Adaptive Fitness Sharing (AFS):** The dynamic niching algorithm extended by a the niche radius adaption mechanism and the the niche-variance reward.

To gather a statistical relevant sample for each configuration, each algorithm is run 75 times on each problem. To check the ability of the niche adaption mechanism, the niche radius for FS and the initial niche radius for AFS is varied in a range from 0.75 to 1.0. This setup creates a huge amount of data – far too much to discuss it in detail within this study. The focus of this study is set on applying fitness sharing to maintain a high population diversity and therewith counteract premature convergence. This should now become evident from the fitness of the best individuals found for each configuration. Thus, for each configuration $c$ the best individual for each run is grouped in a set $S_c$. This set of 50 individuals is later employed to rate the performance of the configuration. A Wilcoxon rank-sum test is applied to compare two sets $S_{c1}$ and $S_{c2}$ for the three hypothesis: $H_0$: $S_{c1}$ and $S_{c2}$ come from the same distribution, $H_1$: $S_{c1}$ contains fitter individuals than $S_{c2}$ ($S_{c1} \leq S_{c2}$), and $H_2$: $S_{c2}$ contains fitter individuals than $S_{c1}$ ($S_{c1} \geq S_{c2}$). The $p$-value as a result of the Wilcoxon rank-sum test can be interpreted as the the probability for $H_0$ to be true, given the data $S_{c1}$ and $S_{c2}$ and considering the alternative hypothesis $H_a$. Therefore, a $p$-value lower or equal $p_s = 0.05$ indicates, that $H_0$ can be rejected on the 0.05-significance level.

### A.2.2 Results

The resulting $p$-values of the Wilcoxon rank-sum tests for each hypothesis and considered pairing of configurations are listed in Tab. A.2. It turns out that FS significantly outperforms the reference configuration NS in only 7 (8 for $p_s = 0.1$) out of 48 test cases. In the contrary, there are 14 setups where NS finds significantly better solutions over FS, i.e. where FS fails (for some $\sigma_s$ in F3, F4, C1, C2, and C4). This may be founded in FS's reduction of the exploitative mechanisms of the evolutionary search due to the penalty for densely populated but none the less promising regions in the search space. An interesting observation is the significant improvement of FS over NS for the convex benchmark function F1: there, it can be assumed, that premature convergence occurs due to a collapse of the population to a subspace of $X$. The adaptive sharing algorithm (AFS) finds in 23 (26 for $p_s = 0.1$) cases a significantly fitter solution than NS. There is no indication of significance for the opposite hypothesis that NS would find fitter solutions than AFS in any case. In 31 (32) cases AFS significantly outperforms even the FS configuration. For 6 (7) cases, the performance in terms of the best solution
found is better for FS over AFS – all in unconstrained problems (F1,F2), by
trend for lower values of $\sigma_s$. 
<table>
<thead>
<tr>
<th>Name</th>
<th>n</th>
<th>Objective and Constraints</th>
<th>(x)</th>
<th>(\mathcal{F}(x^*))</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>10</td>
<td>[F(n, x) = \sum_{i=1}^{n} x_i^2] (x_i \in [-10.0, 10.0])</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>F2</td>
<td>20</td>
<td>[F(n, x) = \sum_{i=1}^{n-1} 100(x_i - \frac{x_{i+1}^2}{4})^2 + (1 - x_i)^2] (x_i \in [-10.0, 10.0])</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>F3</td>
<td>10</td>
<td>[F(n, x) = -\sum_{i=1}^{n} \sin(x_i) \sin(20 \frac{x_i^2}{\pi})] (x_i \in [0, \pi])</td>
<td>-9.66</td>
<td></td>
</tr>
<tr>
<td>F4</td>
<td>20</td>
<td>[F(n, x) = 10n + \sum_{i=1}^{n} x_i^2 - 10 \cos(2x_1)] (x_i \in [-5.12, 5.12])</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>C1</td>
<td>13</td>
<td>[F(x, y) = 5x_1 + 5x_2 + 5x_3 + 5x_4 + 5x_5 - 5 \sum_{i=1}^{4} x_i^2 - \sum_{i=1}^{9} y_i] (0 \leq x_i \leq 1 \forall i \in {1, 2, 3, 4} \quad 0 \leq y_i \leq 1 \forall i \in {1, 2, 3, 4, 5, 9} \quad 0 \leq y_i \forall i \in {6, 7, 8})</td>
<td>-15</td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>6</td>
<td>[F(x) = -25(x_1 - 2)^2 - (x_2 - 2)^2 - (x_3 - 1)^2 - (x_4 - 4)^2 - (x_5 - 1)^2 - (x_6 - 4)^2] (0 \leq x_1 \leq 300) (0 \leq x_2 \leq 4) (0 \leq x_3 \leq 5) (0 \leq x_4 \leq 6) (0 \leq x_5 \leq 10)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td>7</td>
<td>[F(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_1^6 + 7x_2^8 + \frac{1}{2} 4x_2x_3 + 10x_2 - 8x_3 + \frac{1}{2} 4x_3^2 + 3x_3 + 4x_4^2 + 5x_5 \leq 0] (0 \leq x_2 \leq 4) (0 \leq x_3 \leq 5) (0 \leq x_4 \leq 10) (0 \leq x_5 \leq 10)</td>
<td>680.63</td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td>10</td>
<td>[F(x) = (x_1 - 20)^2 + 10(x_2 - 20)^2 + 3(x_3 - 11)^2 + 2(x_4 - 10)^2 + (x_5 - 1)^2 + (x_6 - 3)^2 x_i \in [-10.0, 10.0]) (0 \leq x_1 \leq 300)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table A.1: Unconstrained (F) and constrained (C) benchmark problems with search space \(X\) and best known solution \(\mathcal{F}(x^*)\). All objectives are to be minimized.
A.3 Conclusions

A similarity concept has been identified as a key component in existing niching methods. In order to directly operate on similarity information, existing fitness sharing mechanisms are reformulated from the usually cited distance-based approach to a similarity formulation. This offers the advantage of normalized parameter settings in a range between zero and one, as well as the possibility to embed problem specific similarity metrics which may be more intuitively formulated for structural optimization problems.

Fitness sharing ideas are extended to a single-objective, non-scalar fitness definition for constrained minimization and maximization problems like the ones typically encountered in structural optimization.

An adaptive dynamic niche identification mechanism for evolutionary algorithms has been developed. The adaption comprises a feedback control rule on the niche radius $\sigma_s$, as well as a varying reward for niches offering potential for further development. Thus, the method is intended to maintain a high number of niches and a high diversity inside of each niche. The method has been investigated on a testbed of eight unconstrained and constrained benchmark functions. Its performance in terms of the ability to find fit solutions is compared to a non-sharing reference configuration and a non-adaptive dynamic sharing algorithm. There, statistically significant improvements can be seen for the adaptive over the two alternative strategies in a majority of all investigated configurations. Moreover, there is no statistically significant performance drawback for the adaptive strategy against the non-sharing reference configuration for any of the investigated settings. Thus, at least for the here analyzed benchmark functions the adaptive strategy can be recommended as the most generally applicable of the three considered algorithms.
Table A.2: $p$-values of Wilcoxon rank-sum tests for the three configurations No fitness Sharing (NS), Fitness Sharing (FS), and Adaptive Fitness Sharing (AFS). $p$-values lower or equal to $p_s = 0.05$ indicates that $H_0$ has to be rejected on the $p_s$ level (these $p$-values are set in **bold**-font).
Appendix B

Material Data
(a) Material 1: unidirectionally reinforced Carbon-Epoxy prepreg

<table>
<thead>
<tr>
<th>$E_{11}$</th>
<th>135 GPa</th>
<th>$E_{22}$</th>
<th>10 GPa</th>
<th>$E_{33}$</th>
<th>10 GPa</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_{23}$</td>
<td>3.8 GPa</td>
<td>$G_{31}$</td>
<td>5 GPa</td>
<td>$G_{12}$</td>
<td>5 GPa</td>
</tr>
<tr>
<td>$\nu_{23}$</td>
<td>0.3</td>
<td>$\nu_{31}$</td>
<td>0.27</td>
<td>$\nu_{12}$</td>
<td>0.27</td>
</tr>
<tr>
<td>$\rho$</td>
<td>1580 kg/m³</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$X_t$</td>
<td>1450 MPa</td>
<td>$Y_t$</td>
<td>55 MPa</td>
<td>$Z_t$</td>
<td>55 MPa</td>
</tr>
<tr>
<td>$X_c$</td>
<td>1400 MPa</td>
<td>$Y_c$</td>
<td>170 MPa</td>
<td>$Z_c$</td>
<td>170 MPa</td>
</tr>
<tr>
<td>$S_{yz}$</td>
<td>90 MPa</td>
<td>$S_{zx}$</td>
<td>90 MPa</td>
<td>$S_{xy}$</td>
<td>90 MPa</td>
</tr>
</tbody>
</table>

(b) Material 2: unidirectionally reinforced Carbon-Epoxy prepreg (high-modulus)

<table>
<thead>
<tr>
<th>$E_{11}$</th>
<th>220 GPa</th>
<th>$E_{22}$</th>
<th>7 GPa</th>
<th>$E_{33}$</th>
<th>7 GPa</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_{23}$</td>
<td>2.9 GPa</td>
<td>$G_{31}$</td>
<td>5 GPa</td>
<td>$G_{12}$</td>
<td>5 GPa</td>
</tr>
<tr>
<td>$\nu_{23}$</td>
<td>0.2</td>
<td>$\nu_{31}$</td>
<td>0.35</td>
<td>$\nu_{12}$</td>
<td>0.35</td>
</tr>
<tr>
<td>$\rho$</td>
<td>1600 kg/m³</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$X_t$</td>
<td>1100 MPa</td>
<td>$Y_t$</td>
<td>50 MPa</td>
<td>$Z_t$</td>
<td>50 MPa</td>
</tr>
<tr>
<td>$X_c$</td>
<td>1100 MPa</td>
<td>$Y_c$</td>
<td>150 MPa</td>
<td>$Z_c$</td>
<td>150 MPa</td>
</tr>
<tr>
<td>$S_{yz}$</td>
<td>75 MPa</td>
<td>$S_{zx}$</td>
<td>75 MPa</td>
<td>$S_{xy}$</td>
<td>75 MPa</td>
</tr>
</tbody>
</table>

(c) Material 3: Bi-axial woven (0,90°) Carbon-Epoxy prepreg

<table>
<thead>
<tr>
<th>$E_{11}$</th>
<th>54 GPa</th>
<th>$E_{22}$</th>
<th>54 GPa</th>
<th>$E_{33}$</th>
<th>6 GPa</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_{23}$</td>
<td>2 GPa</td>
<td>$G_{31}$</td>
<td>2 GPa</td>
<td>$G_{12}$</td>
<td>2 GPa</td>
</tr>
<tr>
<td>$\nu_{23}$</td>
<td>0.3</td>
<td>$\nu_{31}$</td>
<td>0.3</td>
<td>$\nu_{12}$</td>
<td>0.09</td>
</tr>
<tr>
<td>$\rho$</td>
<td>1500 kg/m³</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$X_t$</td>
<td>530 MPa</td>
<td>$Y_t$</td>
<td>500 MPa</td>
<td>$Z_t$</td>
<td>11 MPa</td>
</tr>
<tr>
<td>$X_c$</td>
<td>350 MPa</td>
<td>$Y_c$</td>
<td>340 MPa</td>
<td>$Z_c$</td>
<td>85 MPa</td>
</tr>
<tr>
<td>$S_{yz}$</td>
<td>100 MPa</td>
<td>$S_{zx}$</td>
<td>100 MPa</td>
<td>$S_{xy}$</td>
<td>58 MPa</td>
</tr>
</tbody>
</table>

Table B.1: Homogenized, orthotropic mechanical properties of a selection of composite materials (Young’s moduli $E$, shear moduli $G$, Poisson’s ration $\nu$, specific mass $\rho$), strength values (tension $X_t$, $Y_t$, $Z_t$, compression $X_c$, $Y_c$, $Z_c$, shear $S_{..}$)
Bibliography


Own publications


Curriculum Vitae

**Personal**

Name: David Christian Keller  
Date of Birth: November 2, 1981  
Nationality: Switzerland  
Address: Habsburgerstr. 66  
CH-5200 Brugg

**Education**

Ph.D., Centre of Structure Technologies, *ETH Zürich*  
04/2006 – 03/2010

Diploma thesis, *ETH Zürich*  
Title: *Graph-based Laminate Optimization*  

Study Mechanical Engineering, *ETH Zürich*  
Subjects: Lightweight Structures and Control Theory  
10/2001 – 10/2005

Gymnasium, *Alte Kantonsschule Aarau*  
Matura Typus C  
08/1997 – 07/2001

**Working Experience**

Internship, *Garaventa AG, Goldau*  
Mechanical engineering, safety analysis  